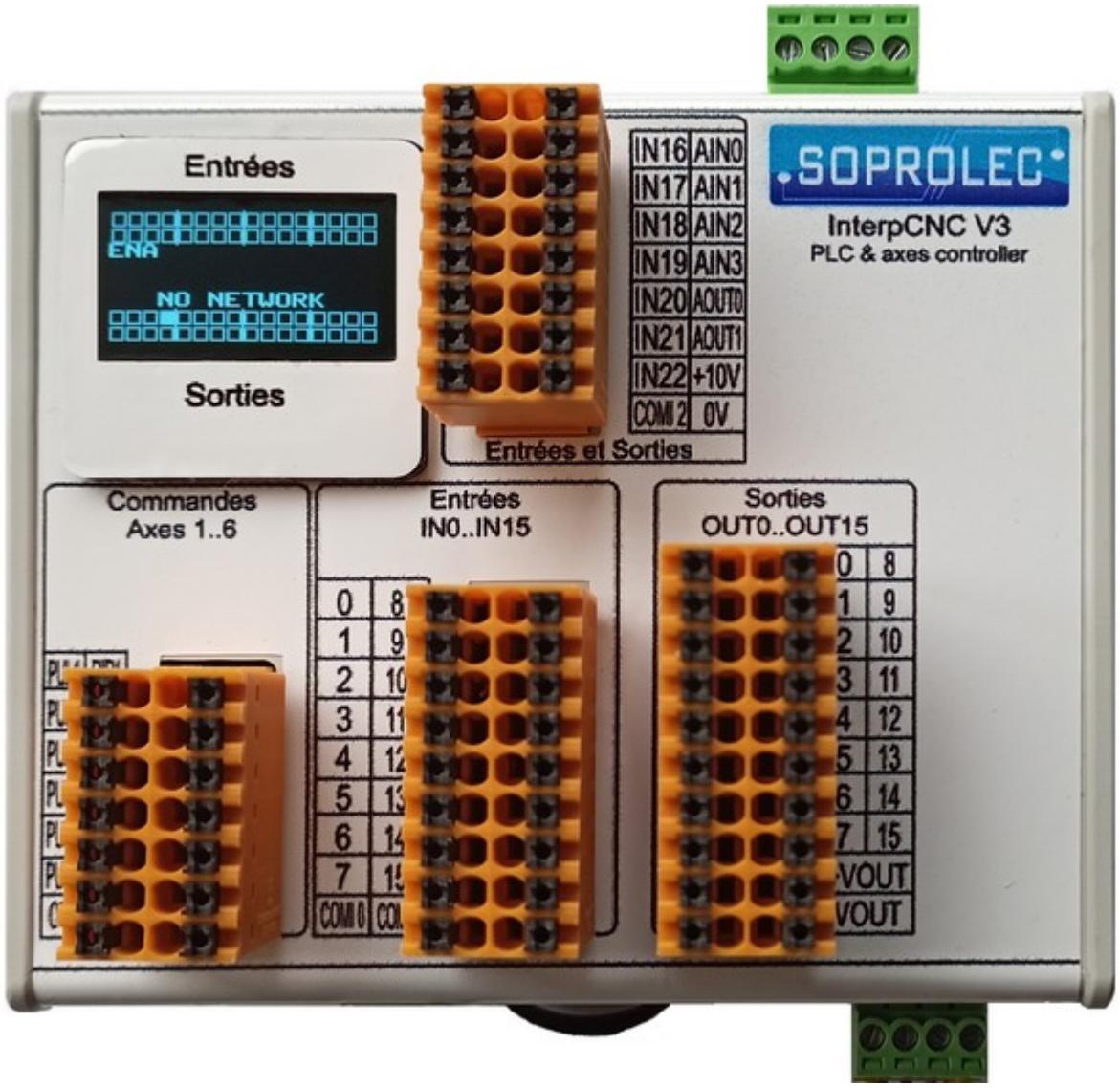
InterpCNC V3
Manuels

Table des matières

Interpreteur_PLC_Basic

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

Carte d'axe
SOPROLEC
InterpCNC V3



Interpréteur langage PLC Basic intégré

Introduction

L'InterpCNC V3 dispose 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).
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).

Cet interpréteur fonctionne en parallèle des autres fonctions de la carte.
Il est donc possible d'utiliser l'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).
La programmation se fait à l'aide du programme ICNCStudio.

Généralités

D’un point de vue matériel, l’InterpCNC 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

Elle a pour avantage d’intégrer un interpréteur PLC Basic, ce qui en fait comme pour toutes les cartes SOPROLEC, l’une des plus accessibles en termes de programmation.
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 êtres manipulés sans perte de précision doivent être compris dans intervalle de ±16777100.

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 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
  OUTPort 0, 0  'Mise à 0 des sorties 0 à 7
  OUTPort 1, 0 'Mise à 0 des sorties 8 à 15
```

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

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, OnTimerEtiq1
 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 V3 reprend bien entendu l'ensemble des expressions, instructions, opérateurs, commandes et fonctions, ainsi que les fonctions de manipulation des chaines 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:

Expressions Littérales / Variables Utilisateur

Expression Littérales

Les chaînes de caractères sont contenus dans des guillemets, ex : "InterpCNC".
Les nombres peuvent être décimaux ou représentés par:
&Hnn Hex Literal, ex : &H3C (60)
&Bnn... Binaire Literal, ex : &B00100011 (35)
n.nE+n Scientifique, e.g. 1.6E+4 (16000)

Variables Utilisateur

Les noms de variables commencent par un caractère 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 $.

Opérateurs

Arithmétiques

^ * / Exponentiation, Multiplication, Division
MOD \ Modulus (reste), Division entière
+ + - Addition, Concaténation de chaîne, Soustraction

Logiques

NOT Inverse logique
= <> Égalité, Inégalité
> < Plus grand que, plus petit que
<= or =< Inférieur ou égal à
>= or => Supérieur ou égal à
AND OR Conjonction, Disjonction
XOR Ou Exclusif

Formatage de chaînes

% [flags] [width] [.prec] type
flags: **-** Justifie à gauche
0 Utilise 0 comme caractère de décalage (pas Espace).
+ Le signe + désigne des valeurs positives.
space Espace comme signe, sauf si négatif.

width: nombre minimal de caractères en sortie, moins cause du décalage, plus cause de l'expansion.
.prec: 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 point(.) si utilisé.
Type: **g ou G** format pour la meilleure présentation.
f ou F Format décimal avec point décimal et chiffres
e ou E Format exponentiel

Commandes / Déclarations

Déclaration de tableau :
DIM variable(éléments…)

Contrôle d'exécution

CONTINUE
DO <déclarations> LOOP
DO WHILE expression <déclarations> LOOP
DO <déclarations> 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
SELECT CASE Variable
CASE Value
CASE ValueFrom TO ValueTo
CASE Value IS < Limite
CASE ELSE
END SELECT

Chaînes / Caractères

ASC (str$)
CHR$ (nbr)
FORMAT$ (nbr [,format$])
INSTR ([start,] search$, pattern$)
LEFT$ (str$, nbr)
LEN (str$)
LCASE$ (str$)
MID$ (str$, start [,nbr])
RIGHT$ (str$, nbr)
SPACE$ (nbr)
SPC (nbr)
STRING$(nbr, val|str$)
TAB(nbr)
UCASE$ (str$)
VAL (str$)
INKEY$

Fonctions

Maths / Nombres

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)

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 function
```

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 V3

Fonctions

GetMB(memo)
GetMW(adr)
GetMDW(adr)
GetMI(adr)
GetMDI(adr)
GetMF(adr)

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)
Ain(nbr)
AinV(nbr)
Out(nbr)

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 G**etMB**(*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… *'si le 31ème bit de l'adresse 3100(U32, ici en Ram) est à 1, alors...*
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*
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
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

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

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 1xxxxx 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 soucis 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 5	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)	Retourne la période de temps écoulée depuis la commande Tic (en micro-secondes) Exemple : ? **Toc(**1**)**
Cyclestat(numéro instance de 0 à 15**)**	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 »*

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	Lecture de l'état actuel d'une sortie. Exemple : ? **Out(**15**)** 'Affiche dans le moniteur, l'état de la sortie 15

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)	Détection d'un front montant sur un bit testé. Exemple : if **DFMBit**(2, GetMB(MBB_ALARME)) then ? « Interruption des cycles sur Alarme » ...
DFDBit(Numero d'instance(1 à 64), Valeur du bit testé)	Détection d'un front descendant sur un bit testé. Exemple : if **DFDBit**(3, GetMB(MBB_ALARME)) then ? « Acquitement des Alarmes »

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'interprêteur 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 chaines
PRINT C$
-> Affichera alors 123456789
```

Fonction	Description	Exemple

ASC(chaine_de_caracteres**)**	Retourne le code ASCII (décimal) du (ou du premier) caractère de la chaîne.	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.	PRINT BIN$(15) → Retourne 1111
CHR$(code_ascii**)**	Retourne le caractère correspondant au code ASCII.	PRINT CHR$(68) → Retourne D
HEX$(nb_entier**)**	Retourne la valeur Hexadécimale correspondant à la partie entièredu nombre.	PRINT HEX$(65535) → Retourne FFFF
INSRT(longueur, chaine1, chaine2**)**	Recherche tout ou partie d'une chaîne de caractères dans une autre, et retourne la position du début.	A$ = "Learn PLC-BASIC" B$ = "BAS" C = INSTR(2, A$, B$) PRINT "Le mot BA commence à la position "; C → Retourne 11
LCASE$(chaine_de_caracteres**)**	Retourne la chaîne de caractères transformée en lettres minuscules.	A$="COMPUTER" PRINT LCASE$(A$) Retourne computer
LEFT$(chaine_de_caracteres, longueur**)**	Retourne une portion de la chaîne de caractères de la longueur indiquée, à partir de la gauche.	A$ = "INTERPCNC" L$ = LEFT$(A$, 6) PRINT L$ → Retourne INTERP
LEN(chaine_de_caracteres**)**	Retourne la longueur de la chaîne de caractères.	A$ = "INTERPCNC" L = LEN(A$) PRINT L → Retourne 9
MID$(chaine_de_caracteres, position, longueur**)**	Retourne une portion de la chaîne de caractères à partir de la position indiquée, pour la longueur donnée.	A$ = "INTERPCNC V3" S$ = MID$(A$, 7, 3) PRINT S$ → Retourne CNC
OCT$(nb_entier**)**	Retourne la valeur octale de la partie entière d'un nombre.	PRINT OCT$(8) → Retourne 10
RIGHT$(chaine_de_caracteres, longueur**)**	Retourne une portion de la chaîne de caractères de la longueur indiquée, à partir de la droite.	A$ = "INTERPCNC" L$ = RIGHT$(A$, 3) PRINT L$ → Retourne CNC

SPACE$(nb_entier**)**	Retourne une chaîne de caractères avec le nombre d'espaces spécifié.	FOR i = 1 TO 6 A$ = SPACE$(i) PRINT A$; i NEXT I
STRING$(repetition, caractere**)**	Retourne une chaîne de caractères contenant le caractère spécifié répété n fois.	PRINT STRING$(2,68) Ou PRINT STRING$(2,"D") → Retourne DD
STR$(nb_reel**)**	Retourne la représentation d'un nombre réel en chaîne de caractères.	N = 453.1 PRINT STR$(n) → Retourne 453.1
TIME$	Retourne le temps écoulé depuis la mise sous tension de la carte. (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$(chaine_de_caracteres**)**	Retourne la chaîne de caractères transformée en lettres majuscules.	A$="computer" PRINT UCASE$(A$) → Retourne COMPUTER
VAL(chaine_de_caracteres**)**	Retourne la valeur numérique située au début d'une chaîne de caractères. Retourne 0 (zéro) s'il n'y en a pas.	N$="1234B5" X=VAL(N$) PRINT "Valeur attendue: "; X → Retourne 1234

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 de gestion du programme

New	Efface le programme PLC Basic en Ram
SaveProgram	Sauvegarde le programme présent en RAM dans une mémoire Flash
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 PLC Basic spécifiques à l' InterpCNC V3

Commandes

SetMB memo, nbr
SetMW adr, nbr
SetMDW adr, nbr
SetMI adr, nbr
SetMDI adr, nbr
SetMF adr, nbr
IncMDW adr[, nbr]
CopyReg adr, nbr

Unlock
Lock
ListFlash
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
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
OUTPort 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_ALARME, 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)*
SetMW Registre, Value	Ecriture dans un registre 16 bits non signé (U16). Exemple : **SetMW** GCYCLE, 10 'écrit la valeur 10, à l'adresse nommée GCYCLE
SetMDW Registre, Value	Ecriture dans un registre 32 bits non signé (U32) Exemple : **SetMDW** *EE_COMPTEUR_TOTAL*, **GetMDW**(*EE_COMPTEUR_TOTAL*)+1 '*incrémentation de 1 du registre EEprom stockant la valeur du compteur total*
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 : **SetMI** 3010, -999999 *'écrit la valeur -999999, à l'adresse 3010 (ici définie en I32, en Ram)*

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)*
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 1er 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)'*

Commandes spécifiques au Hardware de la carte InterpCNC 6 axes

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
SaveProgram	Sauve le programme PLC Basic actuellement en Ram, dans la mémoire Flash (même action que le click sur l'icone Carte SD)
LoadProgram	Charge le programme de l'éditeur PLC Basic dans la Ram.
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	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 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 AxeID, Accel, Vitesse, Decel, Position AxeID : Identifiant de l'Axe (1 à 6) Accel : Accélération en Hz/s Vitesse : Fréquence en Hz des pulses Decel : Décélération en Hz/s Position : Cible, en pas moteur	Met l'axe en mouvement jusqu'à une position cible. Exemple : **MoveAxe** 1, 1500, 10000, 1500, 50936 *'Déplacement X*
MoveSpeed AxeID, accel ou decel (selon la cible), vitesse (signée)	Rotation jusqu'à une vitesse spécifique (vitesse signée) Exemple : **MoveSpeed** 1, 1500, 10000
Home AxeID, InputNumber, InputState, Accel, HighSpeed, Decel, (+/-)MaxStep, LowSpeed, [HomePosition], [MoveStepAfterHome] AxeID : 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	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*

vers le capteur HighSpeed : Vitesse rapide vers le capteur Decel : décélération mouvement rapide vers le capteur (+/-)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)	
Probe AxeID, InputNumber, InputState, Accel, Speed, Decel, (+/-)MaxStep AxeID : 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 Speed : Vitesse vers le capteur Decel : décélération mouvement vers le capteur (+/-)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)
StopAxeID Valeur	Arrêt d'un axe par son Identifiant (numéro où 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	Démarre une capture de temps, en micro-secondes. Exemple : **Tic 1**
CycleStatInit N° d'instance (1 à 5), temps mini, temps actuel, temps maxi	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 êtres 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 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*
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*

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 souhaitions 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

Le temps de latence pour le traitement d'une interruption est < 5µs.
Vous pouvez mettre en place jusqu'à 32 traitements d'interruption 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.

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
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
```

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
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
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
AxeNumber : 1 à 6 (sur Axes motorisés),7 à 9 (sur Entrées codeurs), 10 à 16 (sur Compteurs sur entrées rapides)
CapturePosition : position cible (en pas moteur)
Label : Label de l'interruption
Exemple : **SetCaptureInt** 10, 2, CibleEtiquette, OnPosEtiquette

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 peur 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 (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).

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 : S**etEncoder** Canal, Valeur '*Canal : de 1 à 3.*
Exemple : S**etEncoder** 3, 0 'M*ise 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.
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

COMMUNICATION VIA MODBUS AVEC DES DRIVERS OU VARIATEURS

Chacun des port COM1 et COM2 peut être utilisé pour le pilotage en **MODBUS**, de drivers ou variateurs. Dans ce ce cas, la carte InterpCNC V3 sera « Maître ».
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'autre 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 (-14 = 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

Chaque 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 :

Commande:

```
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                   ' Periode de rafraichissement (ms)
8
9  const ICNC_INPUT_ADDR=1000         ' Addresse modbus de l'état des entrées 0..15 de l'ICNC (Input Register)
10 const ICNC_OUTPUT_ADDR=2160        ' Addresse 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_SOCK_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.

UTILISATION EN MODE DMX

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

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

L'interpCNC vous autorise les 2 modes de fonctionnement :
- Le mode Device disponible sur le port COM1
- Le mode Master disponible sur le port COM2.

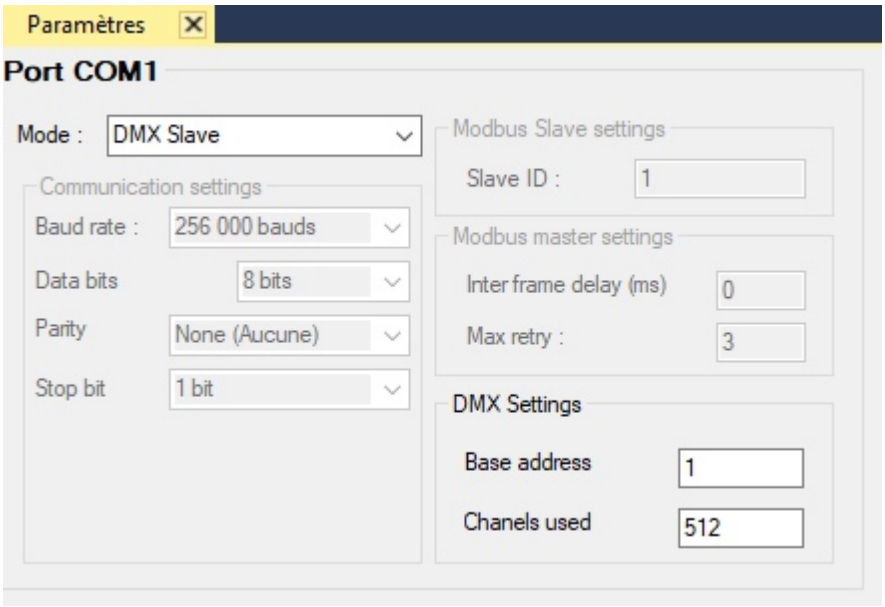
En mode Device, vous pouvez donc piloter les sorties ou des moteurs à partir d'une console DMX.
En mode master, vous pouvez programmer des séquences de contrôle d'appareil en interagissant
avec les entrées de l'automate.

I) Raccordement :

Connecter votre console DMX512 à la carte InterpCNC V3 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 COM1 ou COM2 de la carte InterpCNC.

II) Configuration du mode DMX Device :

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



L'adresse Base est le n° du 1er 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.

III) 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

III) 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 DE L'HORLOGE RTC

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 reglée,
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_SubCommandeCode**, ... 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 nécéssaire car la date sera automatiquement synchronisé.
RTC **RTC_SetAlarmeA**, 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_SetAlarmeA**, 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_StopAlarmeA**
=> 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$(**RTC_GetTimeStr**)
  => Retourne une chaine de caractère représentant l'heure RTC actuelle au format "hh:mn:ss "
  Exemple : ? RTC$(RTC_GetTimeStr); "Message d'information horodaté"

RTC$(**RTC_GetDateStr**)
  => Retourne une chaine de caractère représentant l'heure RTC actuelle au format "jj/mm/aa "
  Exemple : ? RTC$(RTC_GetDateStr); "Message d'information horodaté"

RTC$(**RTC_GetDateTimeStr**)
  => Retourne une chaine de caractère représentant l'heure RTC actuelle au format "jj/mm/aahh:mn:ss "
  Exemple : ? RTC$(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. 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 automatisme.

Plus qu'un outil, ICNCStudio est un véritable studio de développement logiciel (d'où son nom) pour votre carte InterpCNC V3.
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 »).

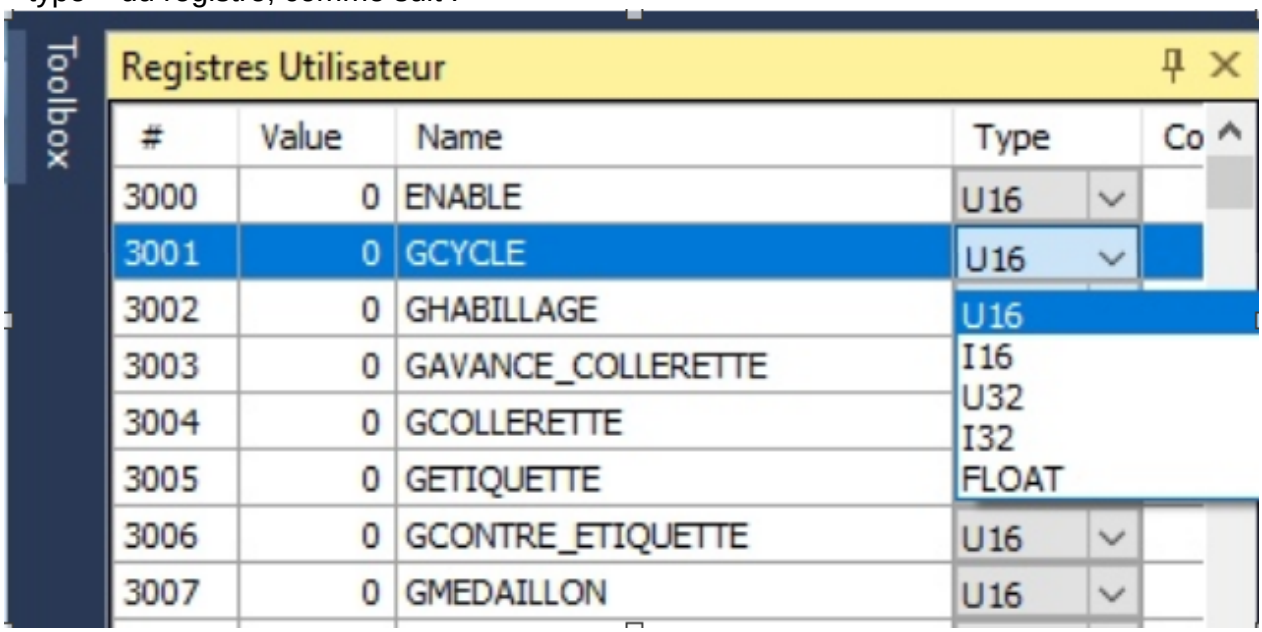
Une aide contextuelle intégrée est disponible en appuyant sur la touche "F1", à tout endroit où se trouve votre curseur,
ou bien pour le texte sélectionné dans l'éditeur (exemple: recherche de la syntaxe pour l'utilisation d'une commande ou d'une fonction.

Registres_Utilisateur

- Fenêtre des **Registres Utilisateur** (RAM)

Ces registres correspondent aux adresses Modbus 3000 à 3999.

Pour chaque Registre Utilisateur , vous définissez directement le typage dans la colonne « type » du registre, comme suit :



U16 et **U32** : entiers 16 et 32 bits.
I16 et **I32** : entiers signés 16 et 32 bits.
FLOAT : nombres flottants.

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier une unité de mesure, etc...)

Les Menus Contextuels :

- **Sur les tableaux de Registres Utilisateur**:
  « Insert Line » et « Remove Line »

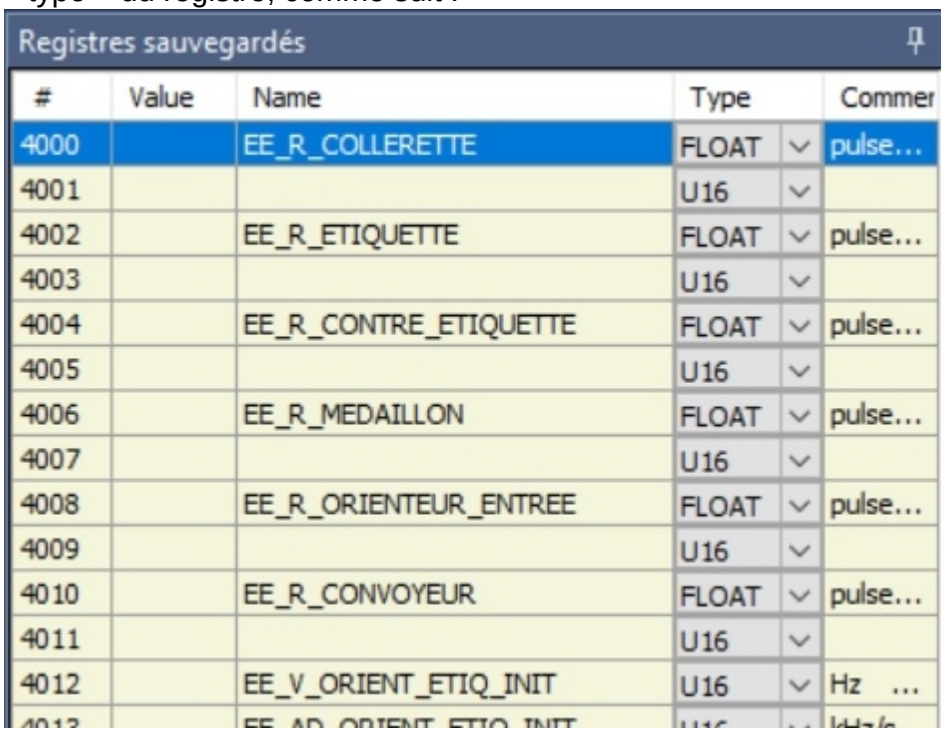
Registres_Sauvegardes

- Fenêtre des **Registres sauvegardés** (=RAM non volatile)

Ces registres correspondent aux adresses Modbus 4000 à 4999.

NB : 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.

Pour chaque Registre Sauvegardé, vous définissez directement le typage dans la colonne « type » du registre, comme suit :



#	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 ...

U16 et **U32** : entiers 16 et 32 bits.
I16 et **I32** : entiers signés 16 et 32 bits.
FLOAT : nombres flottants.

Vous pouvez également pour chaque ligne, insérer un commentaire (par exemple pour spécifier une unité de mesure, etc...)

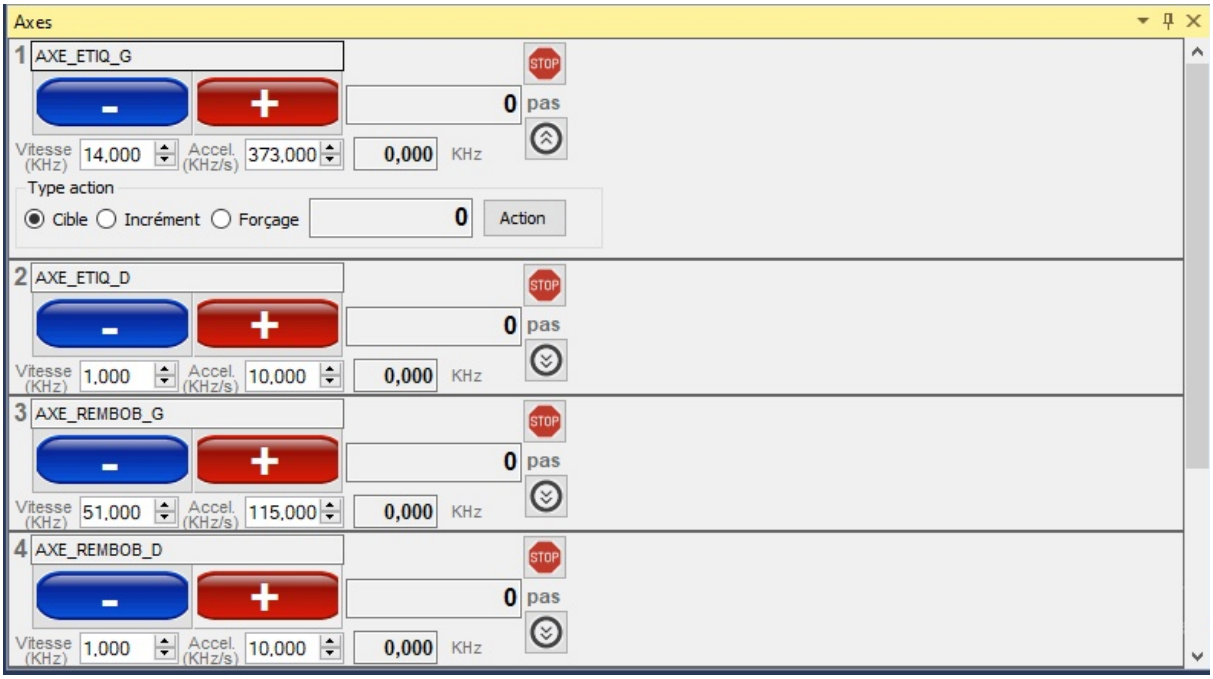
Les Menus Contextuels :

- **Sur les tableaux de Registres Sauvegardés**:
  « Insert Line » et « Remove Line »

Fenetres_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'icone 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.

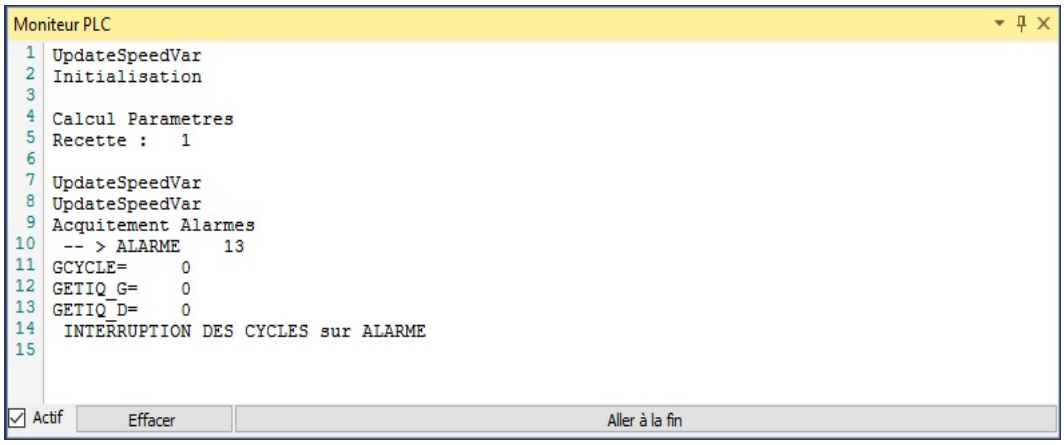
2) Entrer une valeur (en pas, i.e en pulses)

3) Cliquer sur "**Action**"

Le bouton STOP permet d'arrêter immédiatement le mouvement en cours.

Moniteur

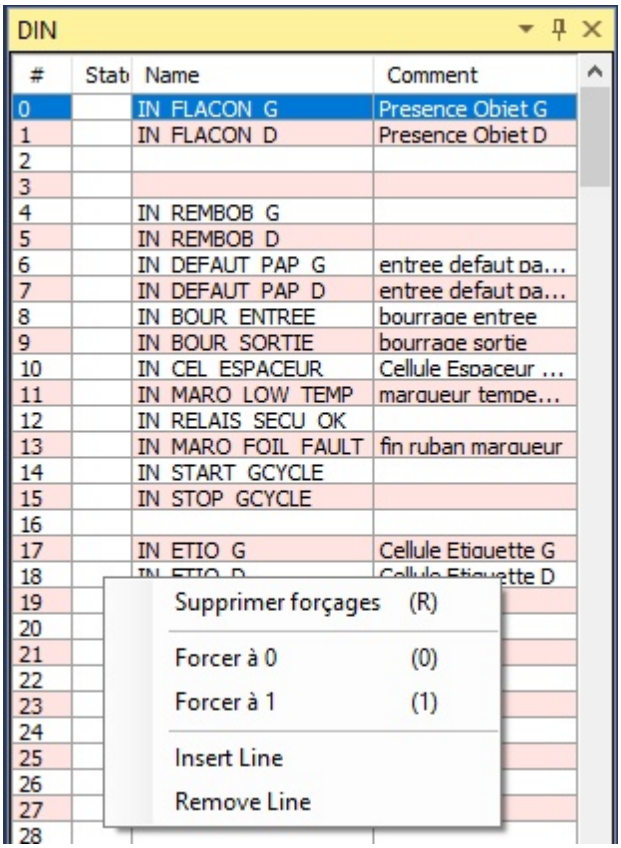
- **Fenêtre du Moniteur** (=Console)
  Dans cette fenêtre s'affichent vos Print (« ? ») en cours d'éxecution du programme, permettant ainsi le debogage,
  ou de suivre le bon déroulement des étapes.



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...)



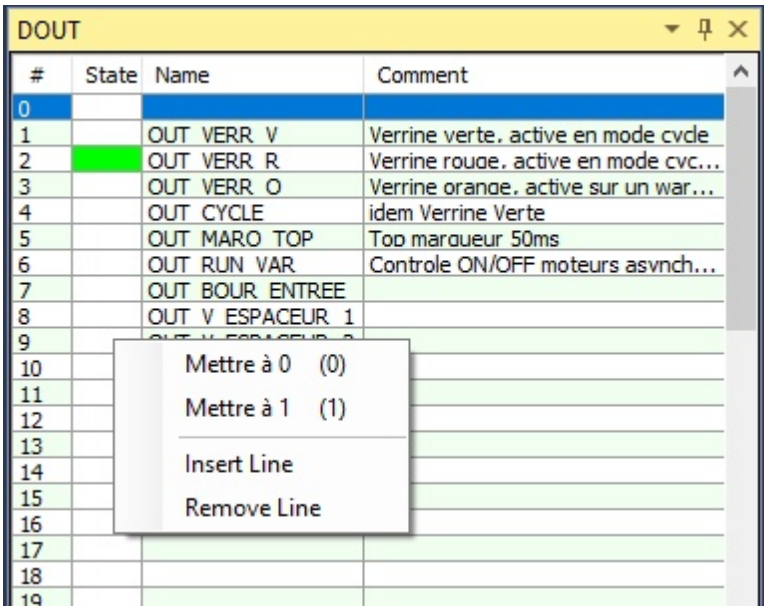
Les Menus Contextuels :

- **Sur le tableau des Entrées** (DIN) :
  « Supprimer Forçage(R) », « Forcer à 0 », « Forcer à 1 », « Insert Line »,
  « Remove Line »

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...)



Les Menus Contextuels :

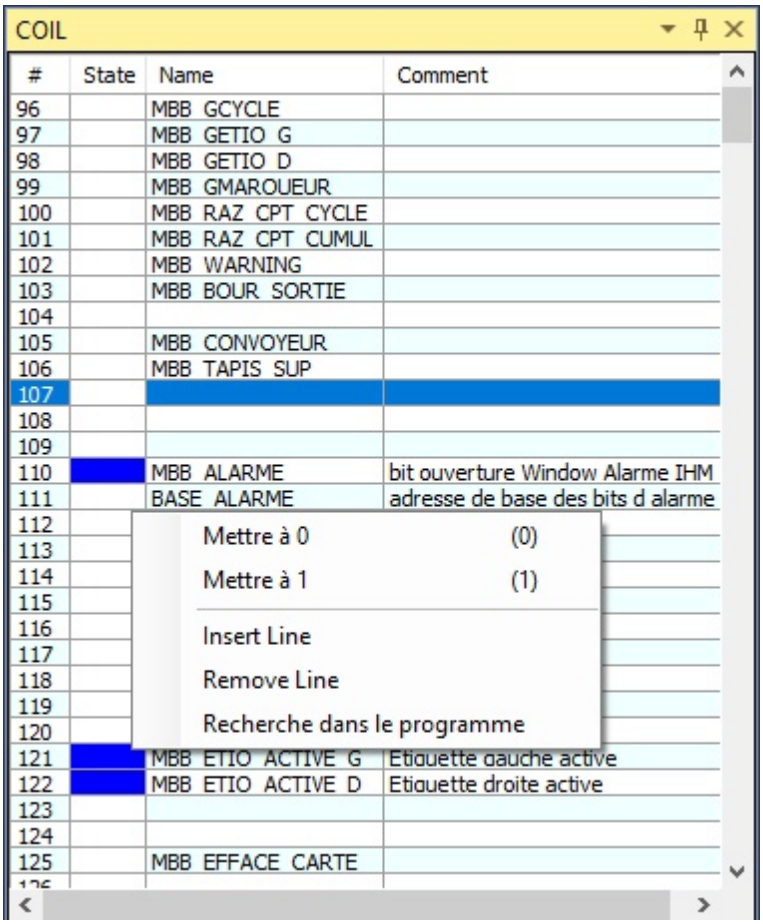
- **Sur le tableau des Sorties** (DOUT) :
  « Mettre à 0 », « Mettre à 1 », « Insert Line », « Remove Line »

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...)



Les Menus Contextuels :

- **Sur le tableau des COILS** (= Memo Bits) :
  « Mettre à 0 », « Mettre à 1 », « Insert Line », « Remove Line », « Rechercher dans le Programme »

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 +/-).

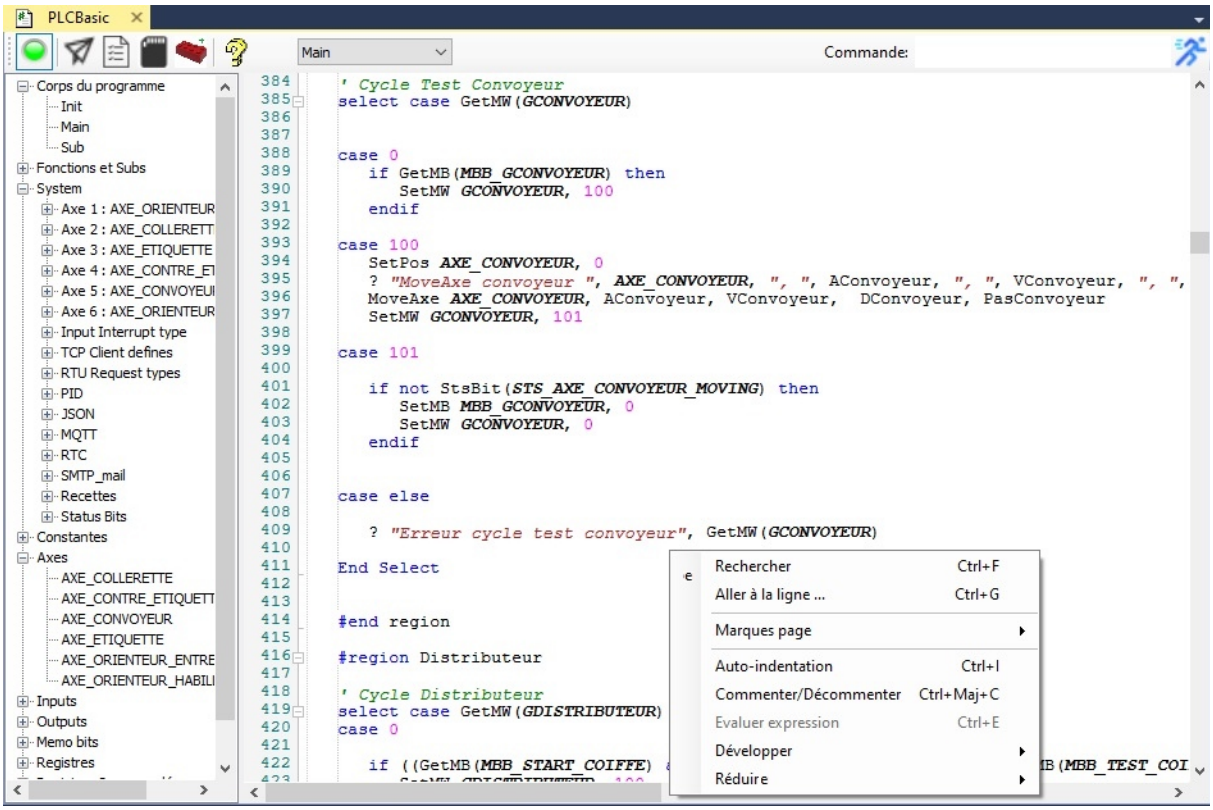
  Depuis cette liste, vous pouvez également faire un "Drag and drop" du nom d'une

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
  → 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**).

Inclusion d'un bloc dans un autre
En tout point de votre programme, que se soit dans le bloc **Main** comme dans tout autre bloc, il est possible d'enchainer l'execution en cours par le contenu d'un autre bloc complet.
Nous utiliserons pour cela l'instruction **Include**

Exemple:
Supposons que nous ayons créé un bloc nommé "Fonctions".

Si dans le bloc Main nous rencontrons:
#Include Fonctions

Alors l'execution du bloc **Main** se deroulera de façon transparente comme si toutes les lignes de code du bloc Fonctions se trouvaient en lieu et place de la ligne "**#Include** Fonctions".

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

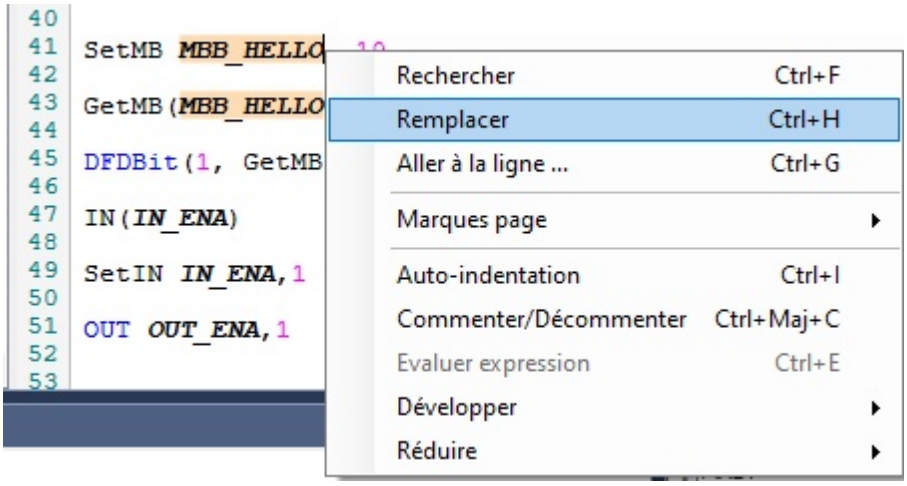
Shift+Ctrl+C : Commenter la zone sélectionnée

Chercher_Remplacer

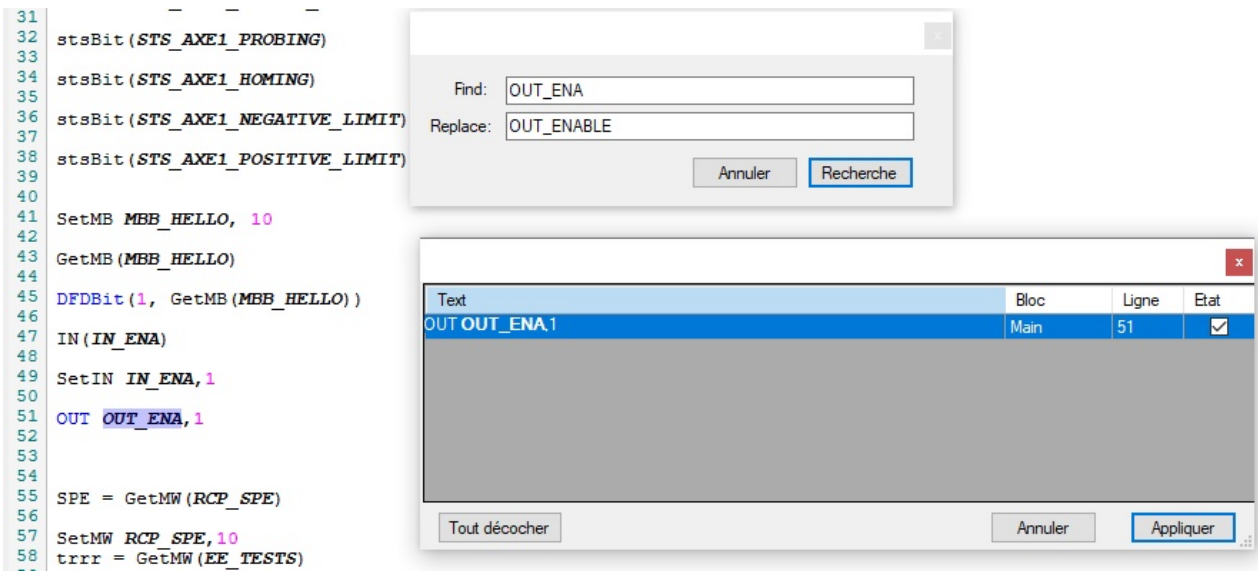
La fonction **Rechercher et Remplacer** (Find and Replace) d' ICNCStudio est très puissante, car elle permet de substituer automatiquement le nom d'une variable de votre programme PLC Basic, ou celui d'un Registre, ou encore celui d'un Bit, par un autre nom et ce pour toutes les occurences et dans toutes 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 raccourcis clavier **Ctrl** + **H** :



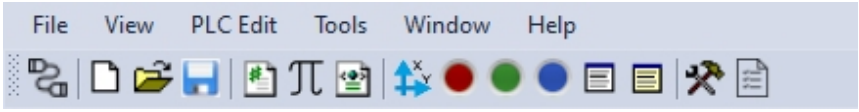
NB: Toutes les occurences rencontrées apparraisent alors dans une fenêtre telle que celle ci-dessus.

Il suffit ensuite de garder cochées les occurences 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

Aperçu de la barre de Menu et la barre d' Outils :



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.

- **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 baarre d'outils. L'option « Graphique » ouvre dans la fenêtre centrale, un graphique permettant de visualiser
  sur des axes Temps (X) et Position (Y), les courbes de vitesse de mouvements manuels lancés depuis la fenêtre « Axes » (choisir l'Axe).

  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.

- **PLC Edit** (Liste des définitions visible, Carte Programme visible, Export programme analysé).
  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 information spécifiques à votre carte : 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 du 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)
→ 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
Dans ce cas les paramètres portent un numéro situé en 20 et 1515.

Description des paramètres les plus utiles

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

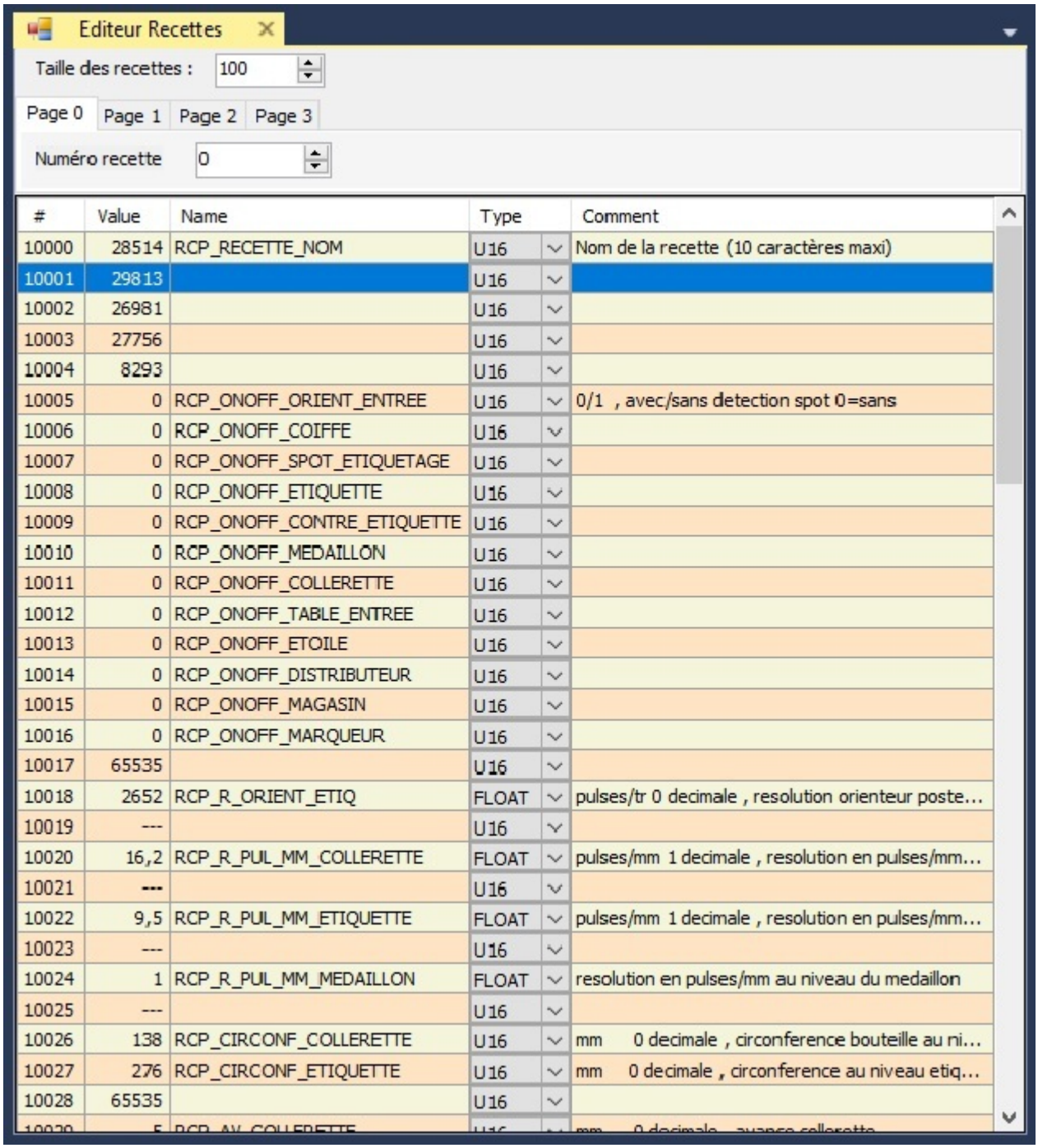
Ici vous pourrez donner un nom Netbios à la carte sur votre réseau, choisir le mode DHCP (à 1), ou bien définir l' adresse IP de votre carte, masque de sous réseau, passerelle, port, etc...

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.



#	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_CONTRE_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	

UTILISATION

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+).
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.

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 : 10000, 11000, 12000 ou 13000.
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.

Variables_PLC

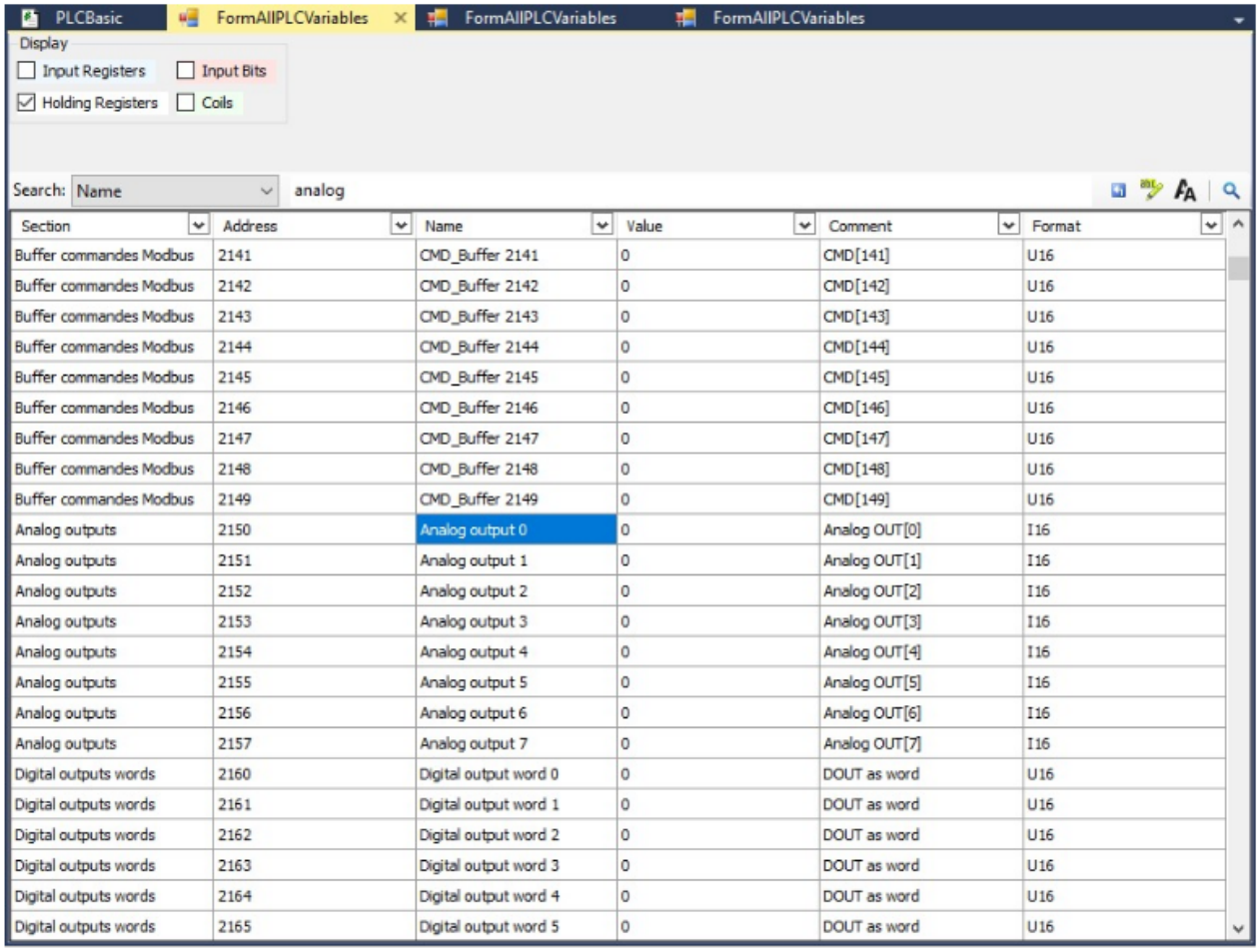
A l'onglet **View -> Toutes variables PLC**

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...)

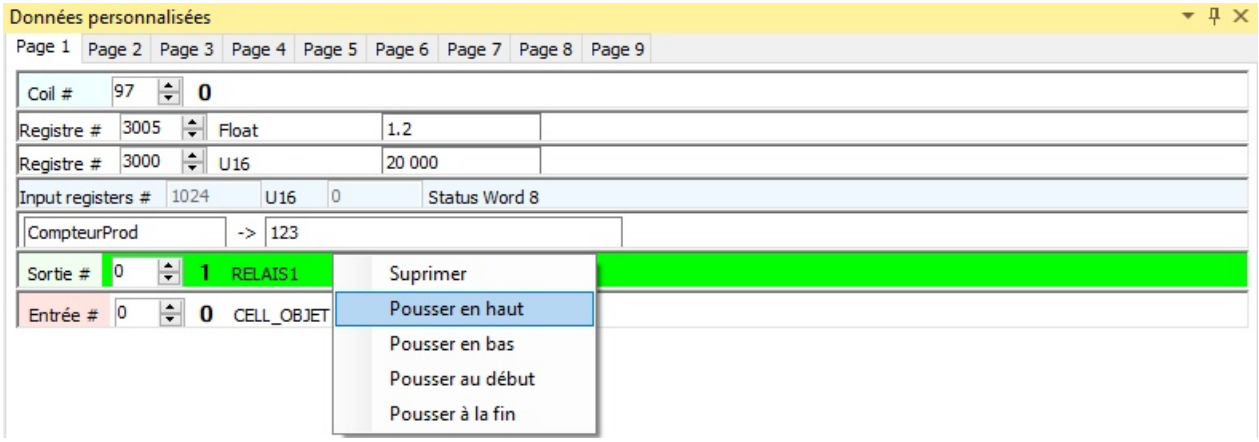
Donnees_personnalisees

Pour déboguer un programme automate en cours de développement, ICNCStudio offre un outil permettant de faire une selection "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.



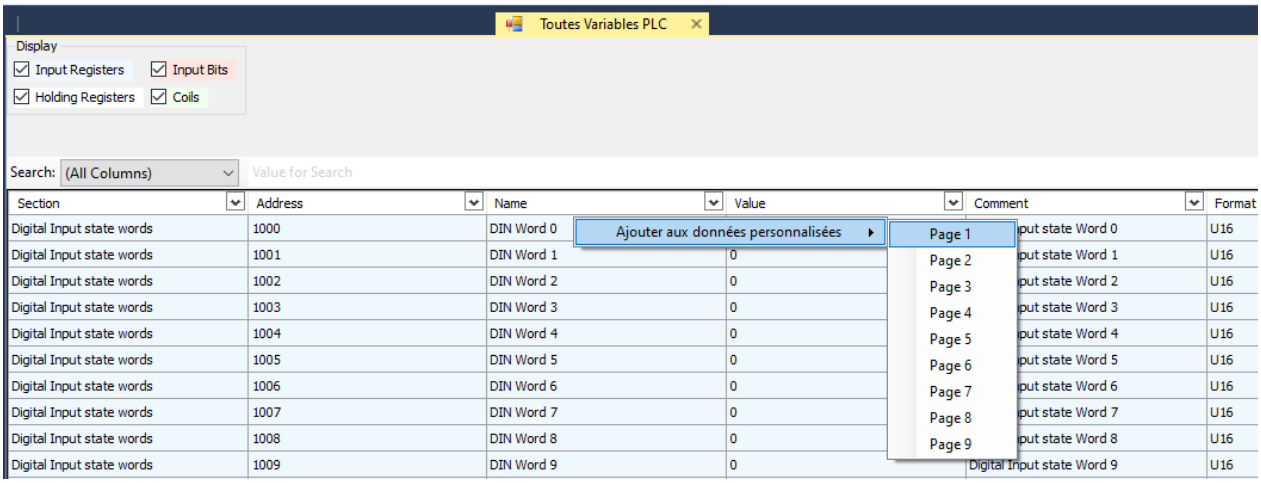
La fenêtre des Données Personnalisées compte 9 pages distinctes (1 à 9), qui peuvent être constituées des registres, bits et variables insérés à votre guise.
Cela est possible depuis les fenêtres **Registres Utilisateurs**, **Registres Sauvegardés**, **DIN**, **DOUT**, **COIL**, **Editeur de Recettes**, 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 au 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

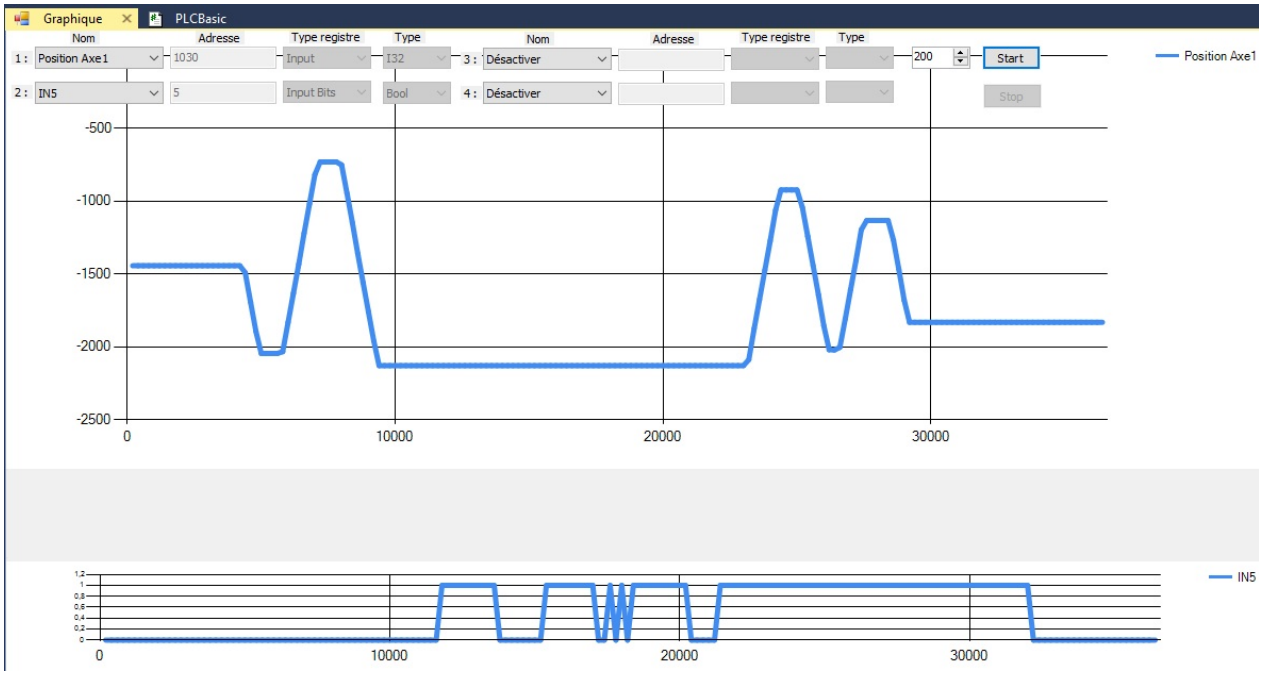
Cette fenêtre permet de monitorer sur une échelle de temps, l'évolution de jusqu'à 4 valeurs, qui peuvent être:

- 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

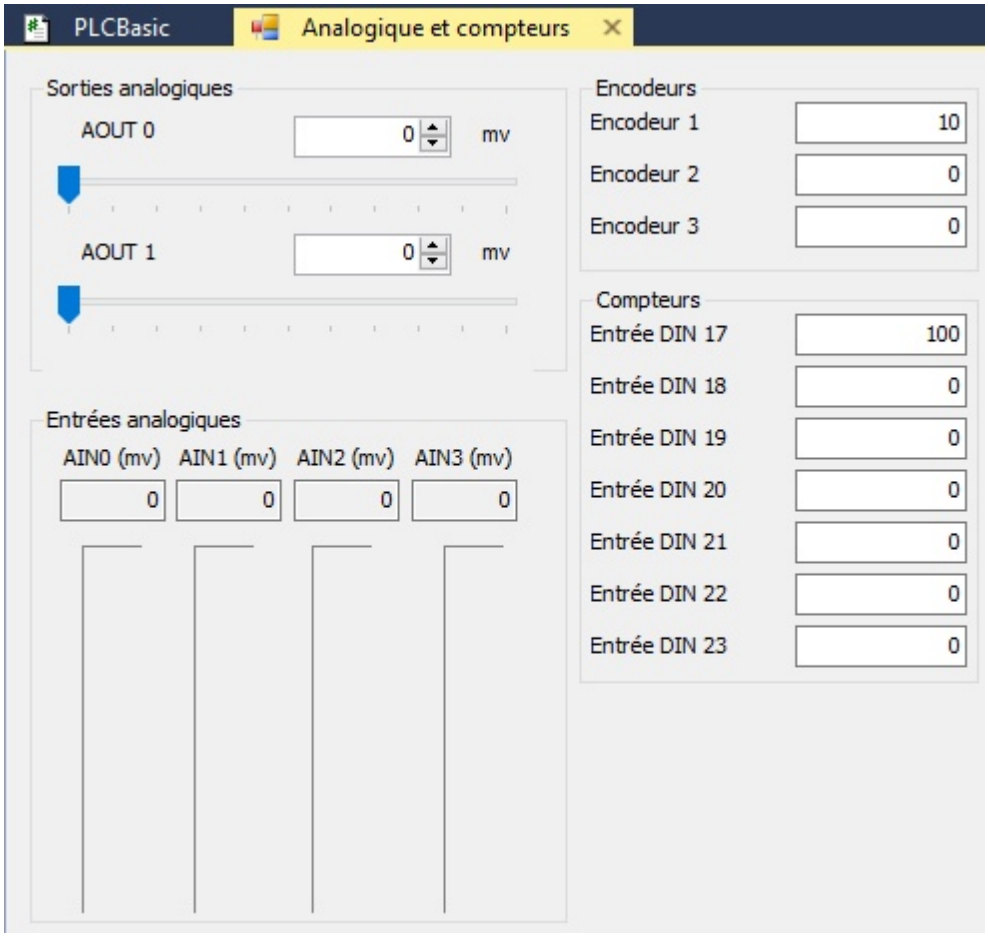
Le délai de rafraichissement est réglable en millisecondes. Réglé à 200ms ici:



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



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

Firmware_update

Mise à jour du firmware de la carte InterpCNC V3

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 pas recommandé de faire une mise à jour.

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, INCNCStudio 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.

Procédure de 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)

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.

ICNCStudio_update

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

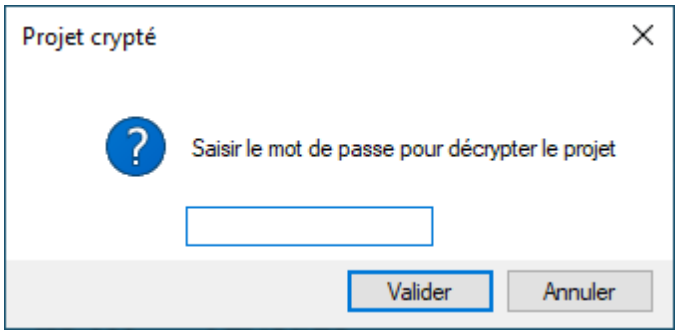
Au delà de la protection par mot de passe, et toujours afin de protéger votre programme

PLC Basic contre la contrefaçon, ICNCStudio intègre aussi un système de cryptage.

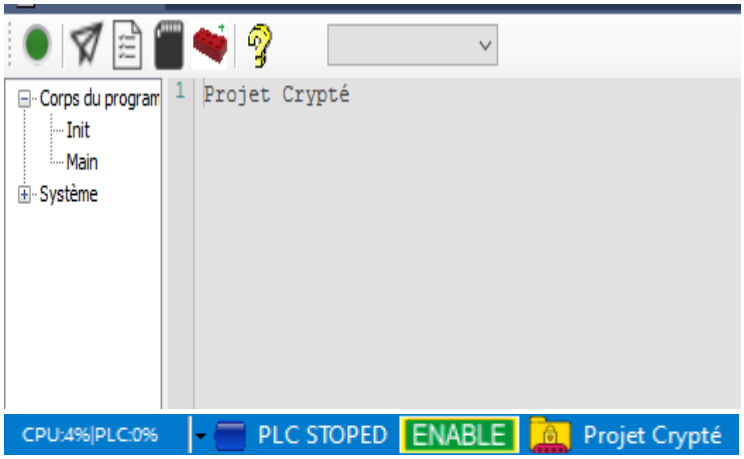
Ainsi un utilisateur avancé qui aurait pû se procurer le fichier source de votre programme automate, ne pourra si vous l'avez crypté, ni en voir le contenu dans ICNCSudio, ni même reconnaitre des portions de code avec un éditeur de texte.

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**

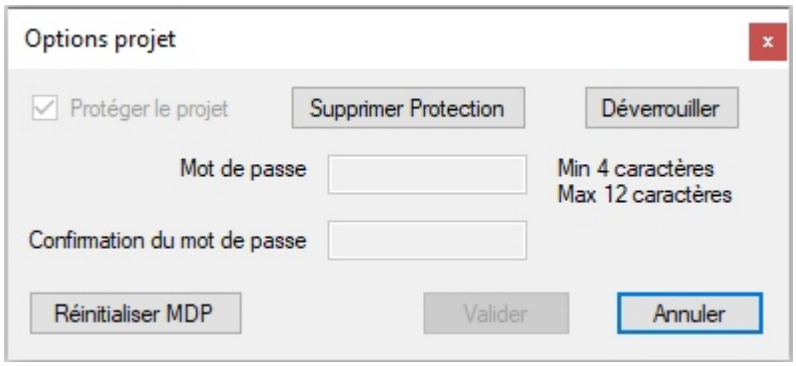
Si vous ouvrez un programme crypté, cette fenêtre de saisie apparaît :



Si vous annulez la saisie du mot de passe, le programme est chargé, mais l'éditeur de texte apparait comme suit:



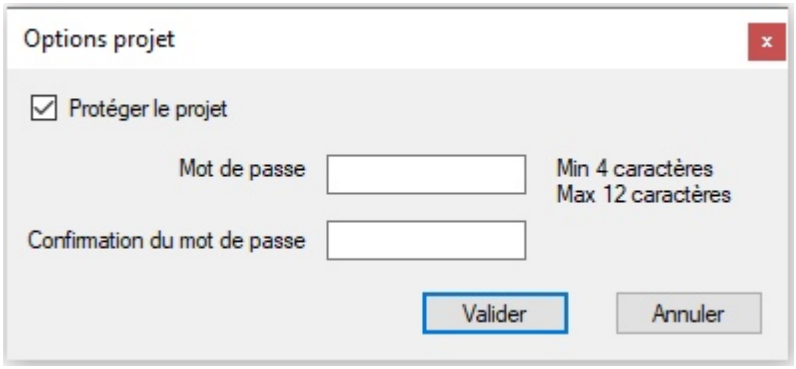
Cliquer sur le logo jaune/rouge (ou bien aller dans **File -> Options Projet)** fera apparaître la fenêtre de saisie du mot de passe de décryptage :



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.

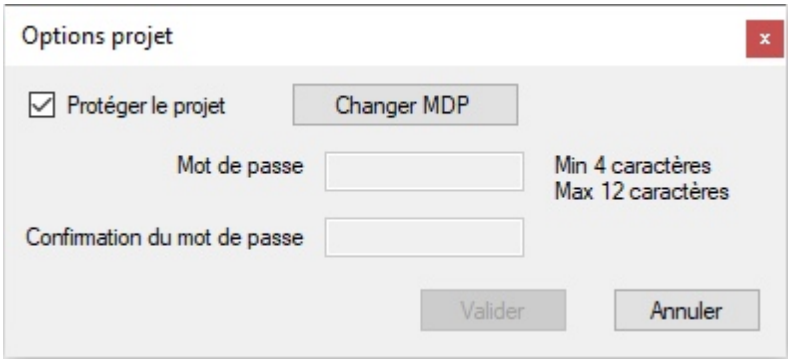
Cliquer sur "**Supprimer Protection**" fera revenir la fenêtre initiale de saisie de mot de passe.
Le contenu du programme devient ensuite totalement accessible:

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 :

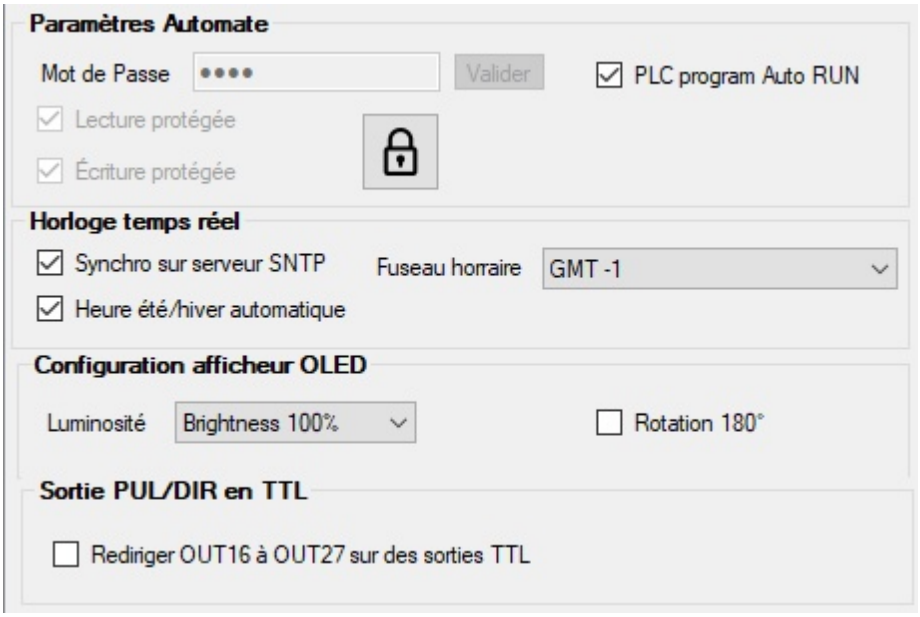


Parametres

Ce chapitre décrit les différents écrans de configuration accessibles depuis l'icone des paramètres.

A la différence du tableau accessible depuis **View** -> **Paramètres InterpCNC,** seuls les principaux paramètres sont proposés,
mais la lisibilité et la selection des réglages est facilitée par des menus déroulants, champs de saisie et les "check box".

Configuration_generale



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.
Ecriture 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.

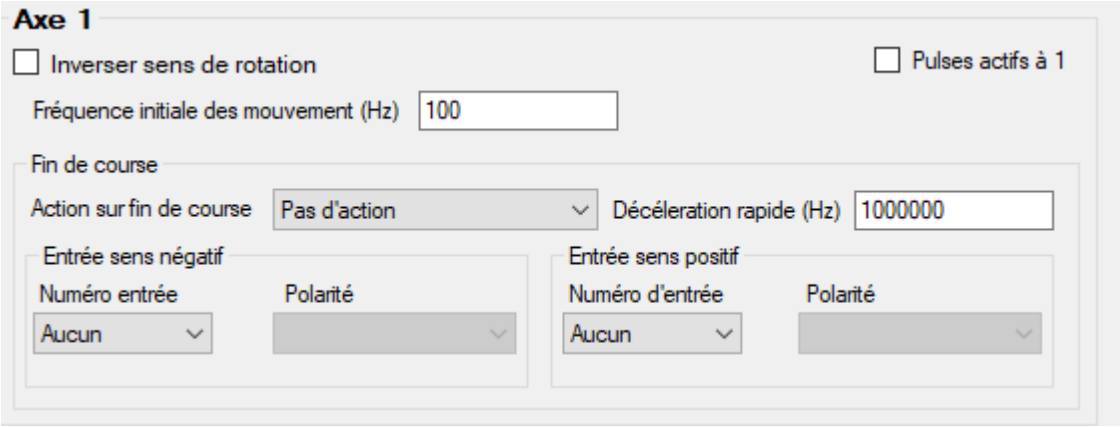
Rotation 180°

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.

Axes



La fréquence initiale des mouvements des Axes

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...)

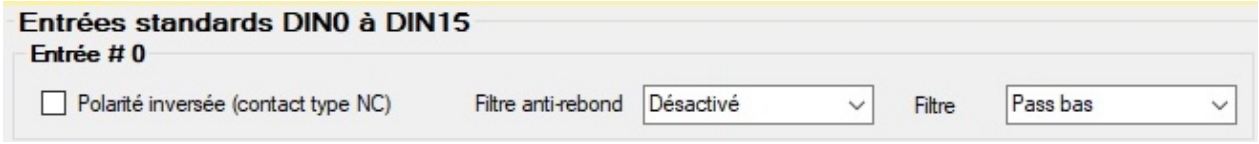
Fin de course

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.

DIN

Configuration des Entrées standards 0 à 15



- **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 :

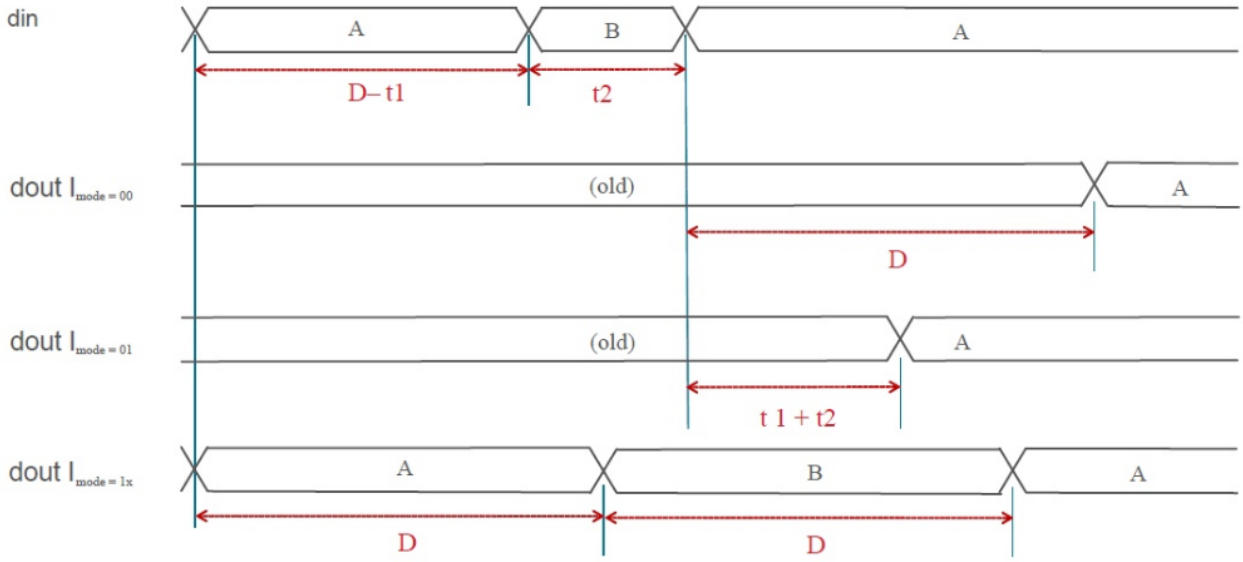
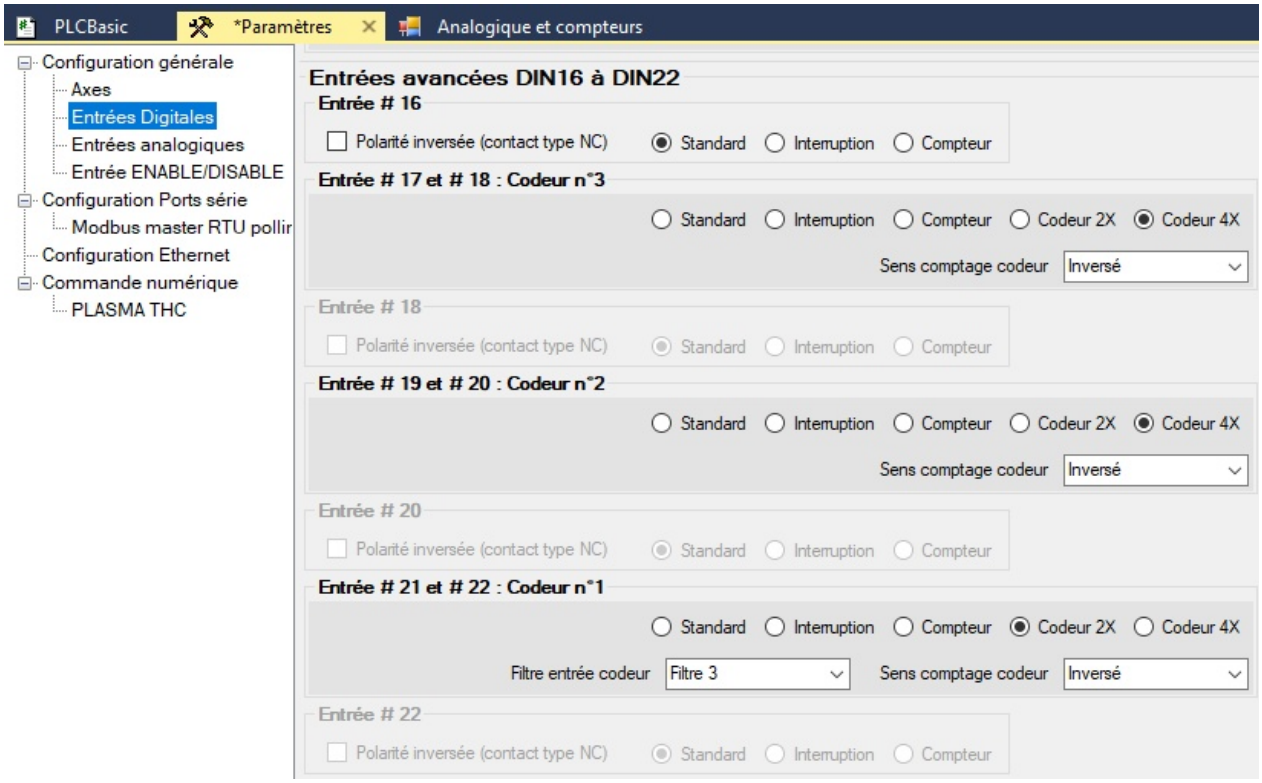


Figure 2.7. Debounce Filter Modes Timing Diagram

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)



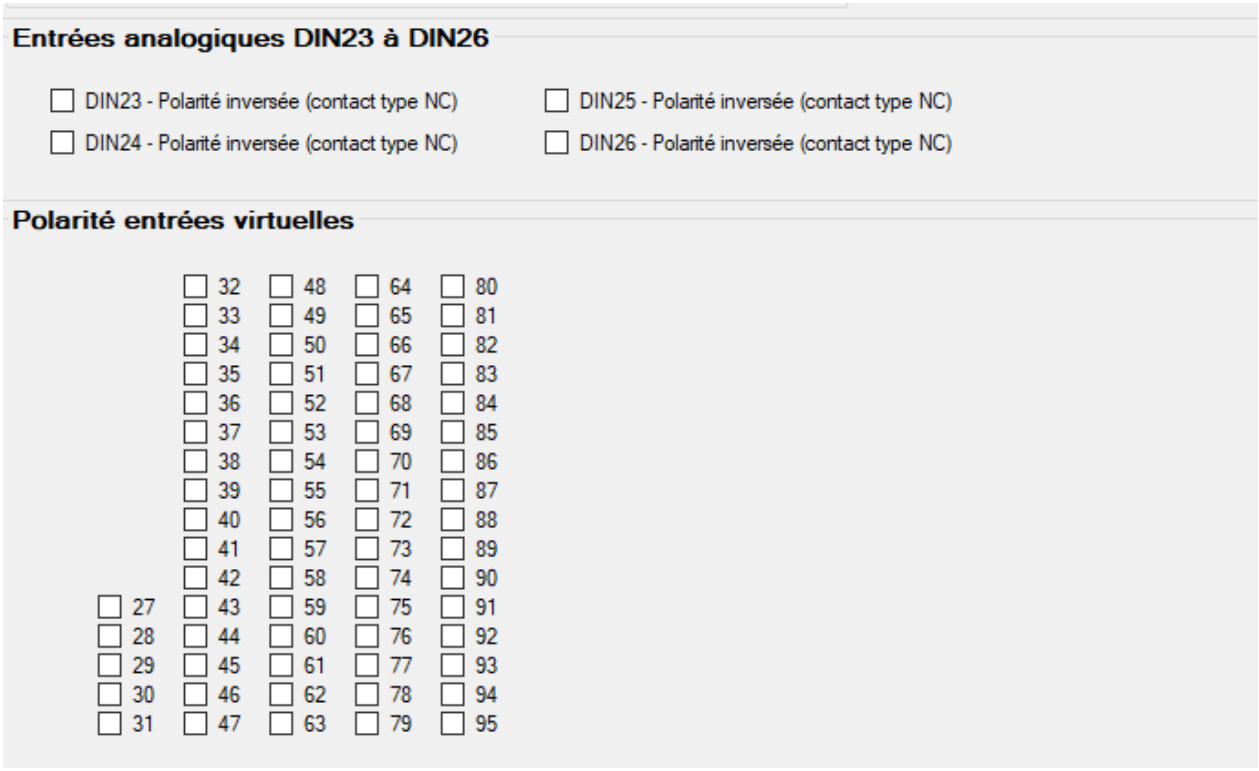
Ces 7 entrées sont configurables comme suit :
0 : rafraîchissement à chaque ligne du programme (Standard).
1 : rafraîchissement à chaque interruption accédant à cette entrée.
2 : mode compteur
3 : mode encodeur 2X
4 : mode encodeur 4X

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.

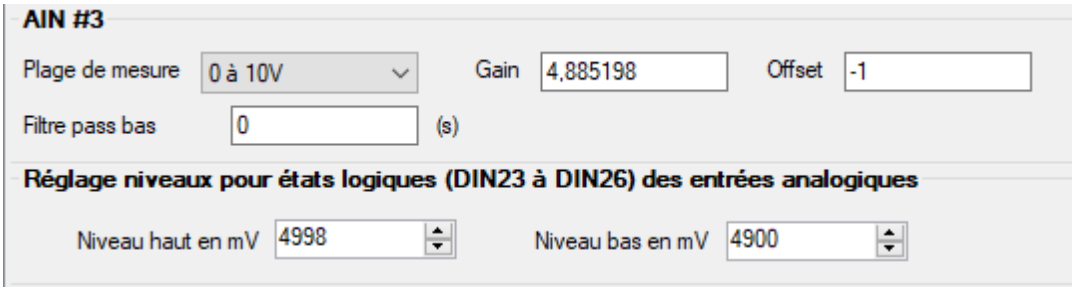
Un réglage de filtrage est possible pour le codeur n°1 (valeur de 1 à 15 selon le niveau du bruit à l'entrée).

La polarité de chacune de ces entrées peut également être inversée ici.

Idem pour les entrées virtuelles, qui peuvent être:
- soit des entrées analogiques utilisées en tant qu'entrées tout ou rien (TOR) et remappée commes entrées numérotées de 23 à 26
- soit des entrées digitales



AIN



- **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.

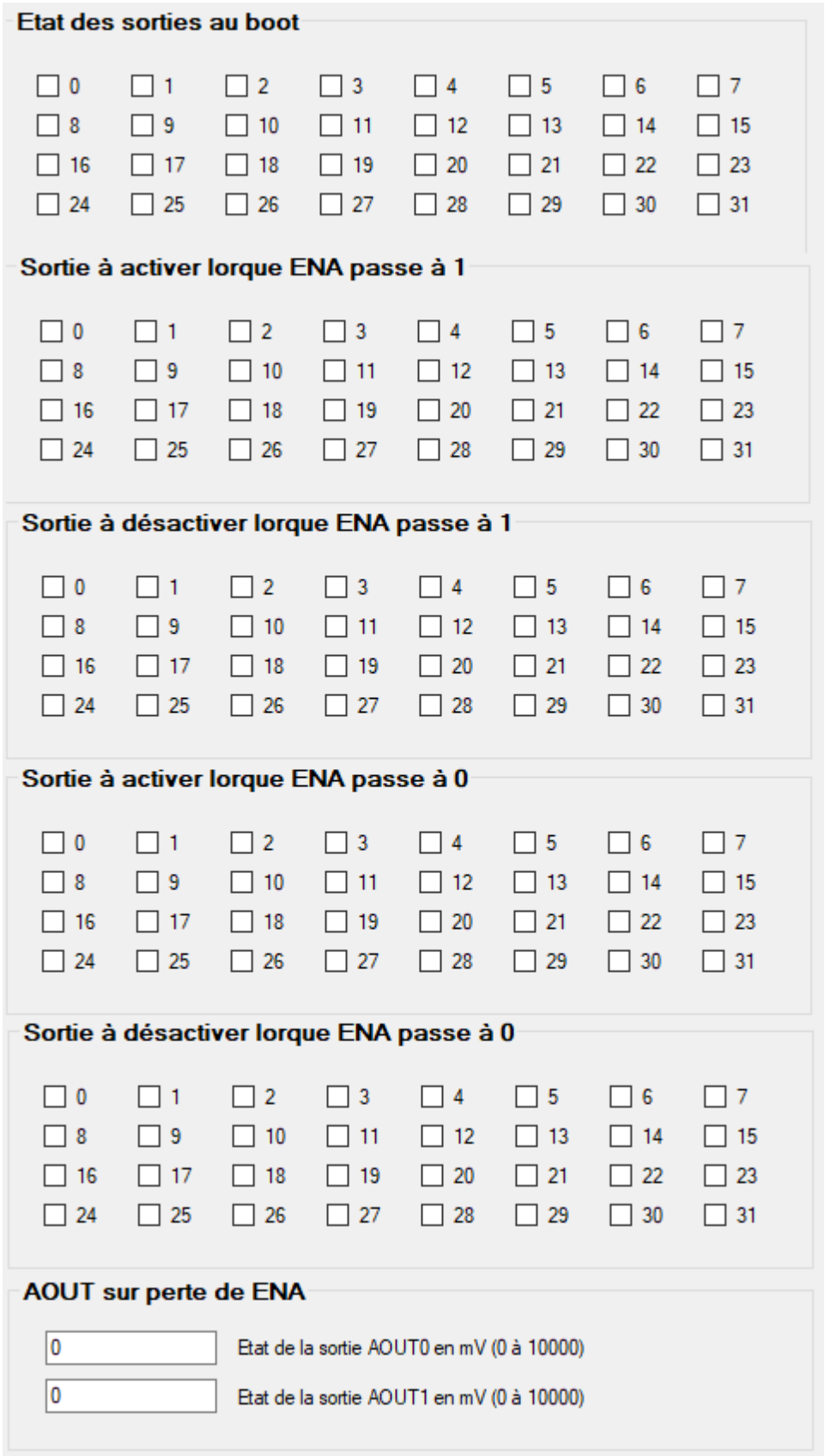
IN_ENA

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.

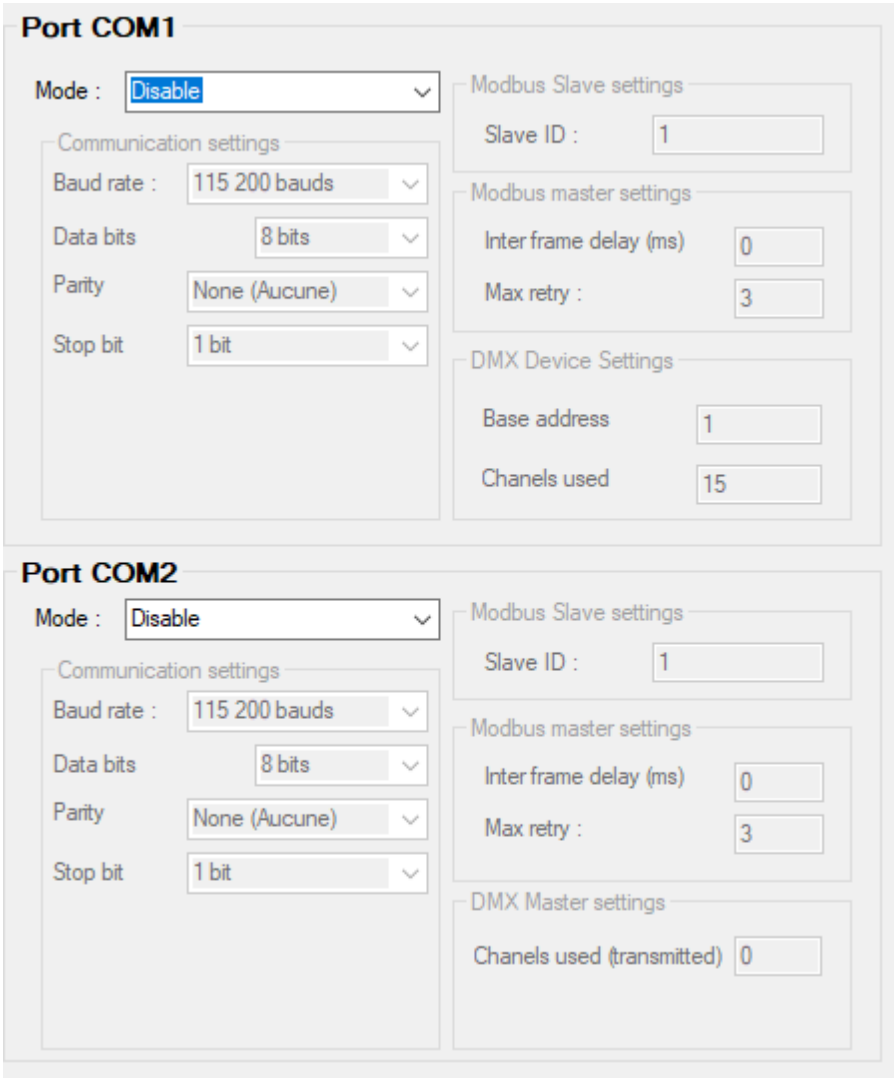
- **L’État des sorties 0 à 31 au boot** (Registre 270)

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)).



- **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



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

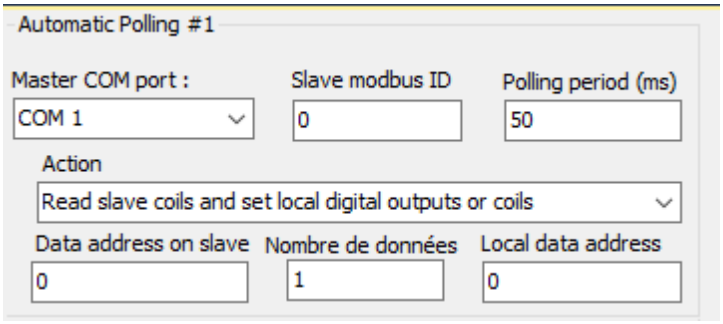
avant de retourner une erreur de transmission (visible dans « Toutes les Variables PLC » -> registres Modbus RTU master error (ou success)).

Polling

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.

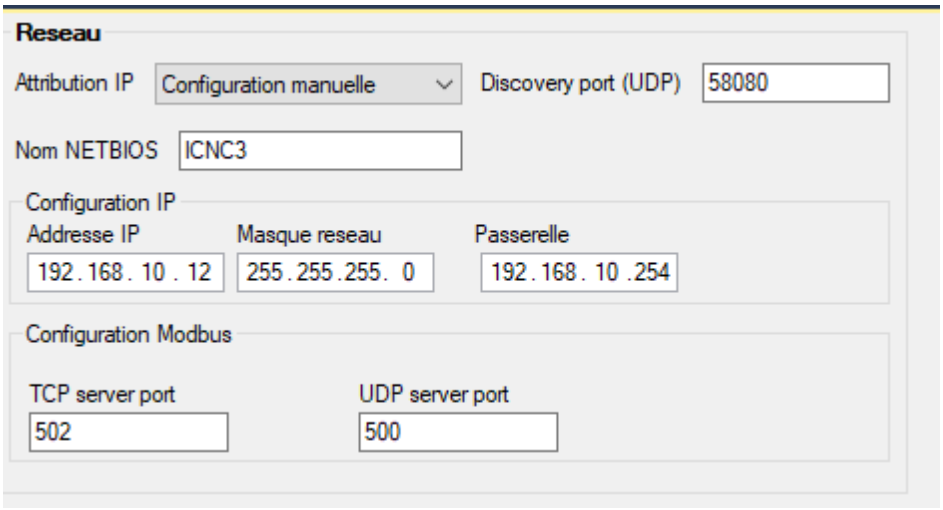
Le polling, appelé aussi mapping (ou mappage en français) consiste à créer une copie auto-actualisée d'une zone de données, vers une autre zone de données.

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.



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**

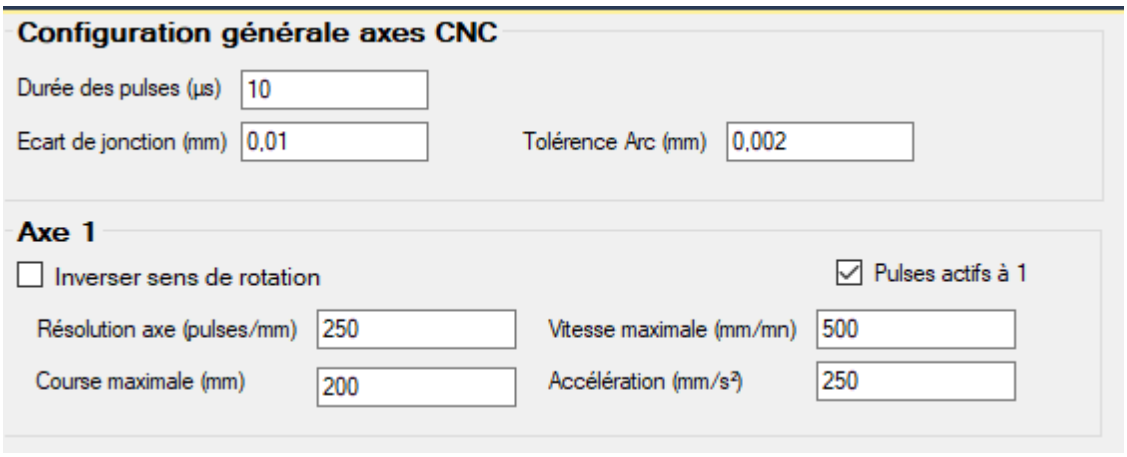
Ethernet



– **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



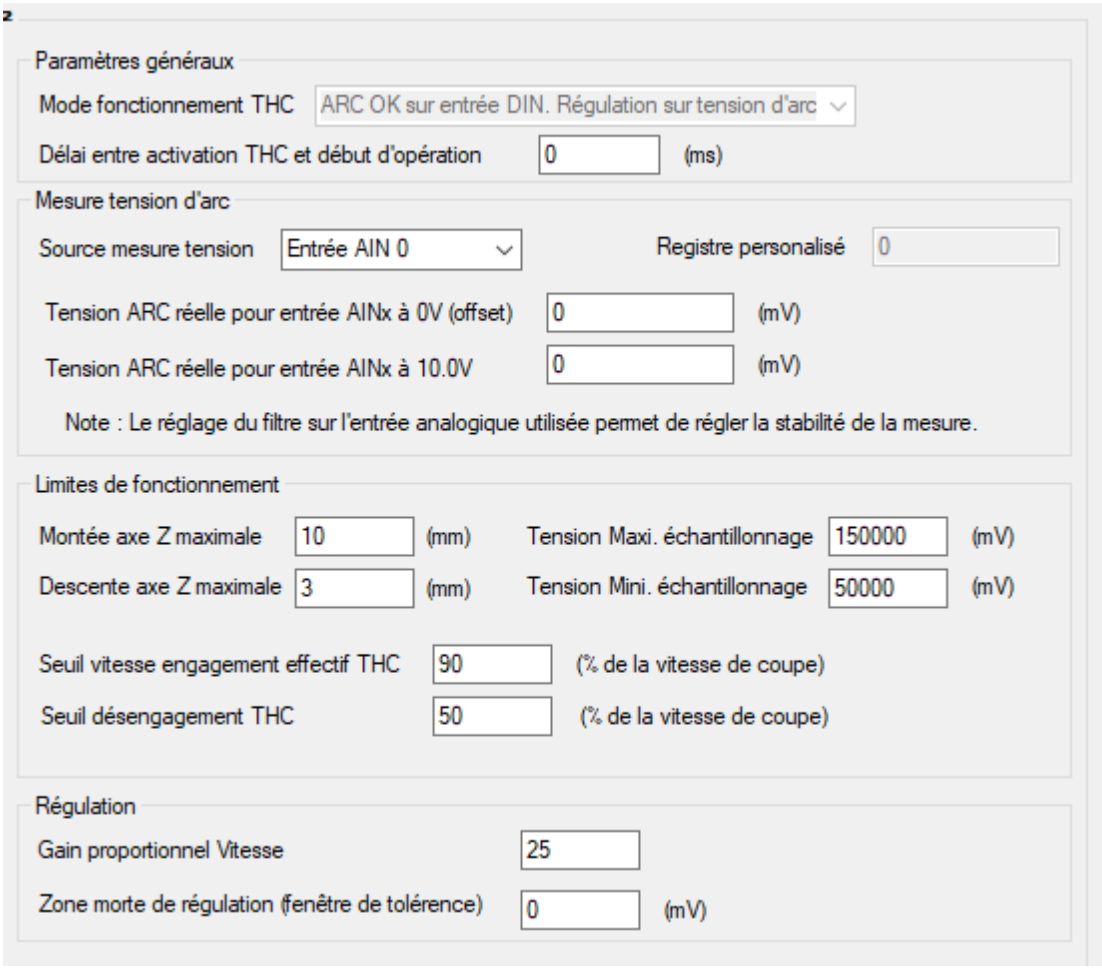
Ces paramètres concernent la configuration avancée des signaux pulses et direction en mode commande numérique, pour chacun des axes.

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.

La carte IntercpCNC V3 est compatible 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.



« **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 InterpCNC. 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.

Remarque : Ces paramètres sont écrits dans ces registres de la carte par le logiciel **GALAAD**. Veuillez lire la section THC du manuel de GALAAD pour une information plus complète.

Release_note

ICNCStudio Version : 1.0.0.87
Corrections :

- Correction d'un bug sur le réglages "Analog, Compteurs": les modifications sur les champs Encodeur 2 et Encodeur 3 n'étaient pas pris en compte
- Correction d'un bug: la fonction Rechercher ne trouvait plus les occurrences dans les blocs restants, après avoir supprimé le dernier bloc du programme automate
- La fonction Search and Replace n'était pas fonctionnelle

Fonctionnalité des Données Personnalisées :

- Correction sur un changement de couleur indésirable lors du changement d'état des Coils
- Autorisation de l'écriture d'un Holding Register depuis le tableau des Données Personnalisées
- Le Text **Modified** ne s'affiche plus dans l'entête d'ICNCStudio suite au chargement d'un programme plc

Évolutions :

Fonctionnalité des Données Personnalisées:

- A l'ouverture de la fenêtre des données personnalisées positionnement automatique en bas
- Changement des bordures + alignement des contrôles
- Ajout d'un bouton Toggle ("Change state") pour les sorties et les coils
- Ajout forçage des entrées (bouton "Forçage à 1", et bouton "Supp Forçage")

Fonctionnalité Search and Replace :

- Redéfinition et application d'un nouveau mode de fonctionnement

ICNCStudio Version : 1.0.0.86

Corrections :
- Réinitialisation de la position de l'onglet "recherche" suite au démarrage d'ICNCstudio.
- Modification de la disposition des paramètres de protection de la carte.
- Élimination du clignotement de l'éditeur de texte suite à la sélection d'un mot dans l'onglet "recherche" (si ce n'est pas nécessaire).
- Élimination du clignotement de l'éditeur de texte suite à la sélection d'une sous-fonction ou d'une fonction dans l'arborescence (si ce n'est pas nécessaire).
- Amélioration générale du fonctionnement du graphique personnalisé.
- Correction du bug de détection de la version du projet chargé dans ICNCstudio.
- Correction de l'erreur d'affichage du numéro de codeur dans l'onglet "analogique et les compteurs".
- Affichage du numéro du codeur suite à un changement des paramètres des entrées rapides.
- Changement des valeurs maximales des filtres sur les entrées analogiques.
- Correction du bug lié au changement constant de la taille du tableau de registres utilisateur et à la sauvegarde des registres.
- Correction lors de l'envoi à la carte, d'un paramètre Codeur (depuis le tableau des paramètres)
- Correction sur la lecture de registres U16 et I16 dans l'affichage du graphique personnalisé.

Évolutions :
- Ajout de fonctions et de commandes pour l'autocomplétion dans l'éditeur de texte.
- Possibilité de réinitialiser le mot de passe d'un projet grâce à un code unique pour chaque projet.
- Navigation dans le tableau de recherche à l'aide des flèches.
- Ajout de la fonctionnalité des Données Personnalisées, pour le débogage.

ICNCStudio Version : 1.0.0.85

Corrections:
- Bug au moment de l'ouverture de la fenêtre de recherche dans l'éditeur PLC et la recherche de caractères spéciaux.

Évolutions:
- Scroll désactivé pour la sélection des recettes.

ICNCStudio Version : 1.0.0.84

Corrections:
- Le curseur reste visible dans l'éditeur basic.
- Modification de l'affichage du nom du projet.
- Copier/Coller des noms de variables à partir des différents tableaux sans récupérer toute la ligne du tableau.
- La taille de la colonne des valeurs dans les tableaux des registres utilisateurs et des registres sauvegardés est stable, ce qui signifie qu'un changement constant de taille à chaque nouvelle valeur est éliminé.

Évolutions:
- Les cellules des tableaux deviennent grises en cas de déconnexion de la carte.

- Pour ouvrir le projet, vous pouvez simplement faire glisser et déposer le fichier dans l'éditeur de texte.
- Ajout de nouveaux paramètres pour le verrouillage de la carte (protection de la lecture/écriture du programme automate sauvegardé).
- Ajout de nouveaux paramètres pour l'interface Linky.
- Numérotation continue des lignes du programme entre les différents blocs.
- Vérification des conflits des noms identiques de variables dans les différents tableaux.
- Vérification des conflits des noms identiques de constantes dans l'éditeur basic avant l'envoi du programme à la carte.
- Optimisation des messages d'erreur, ajout de texte HTML qui renvoie vers le bloc et la ligne d'erreur.
- Nouveau système d'aide contextuelle pour l'éditeur basic (recherche de commandes) et pour les différentes rubriques d'ICNCStudio en appuyant sur la touche F1 sur l'élément ou le texte recherché.
- Ajout du cryptage/décryptage des projets.
- Détection des fonctions et des subroutine (sub) dans l'arborescence.
- Nouvelle fenêtre de recherche dans tous les blocs en appuyant sur (Control + F) sur l'éditeur Basic ou sur les différents tableaux.

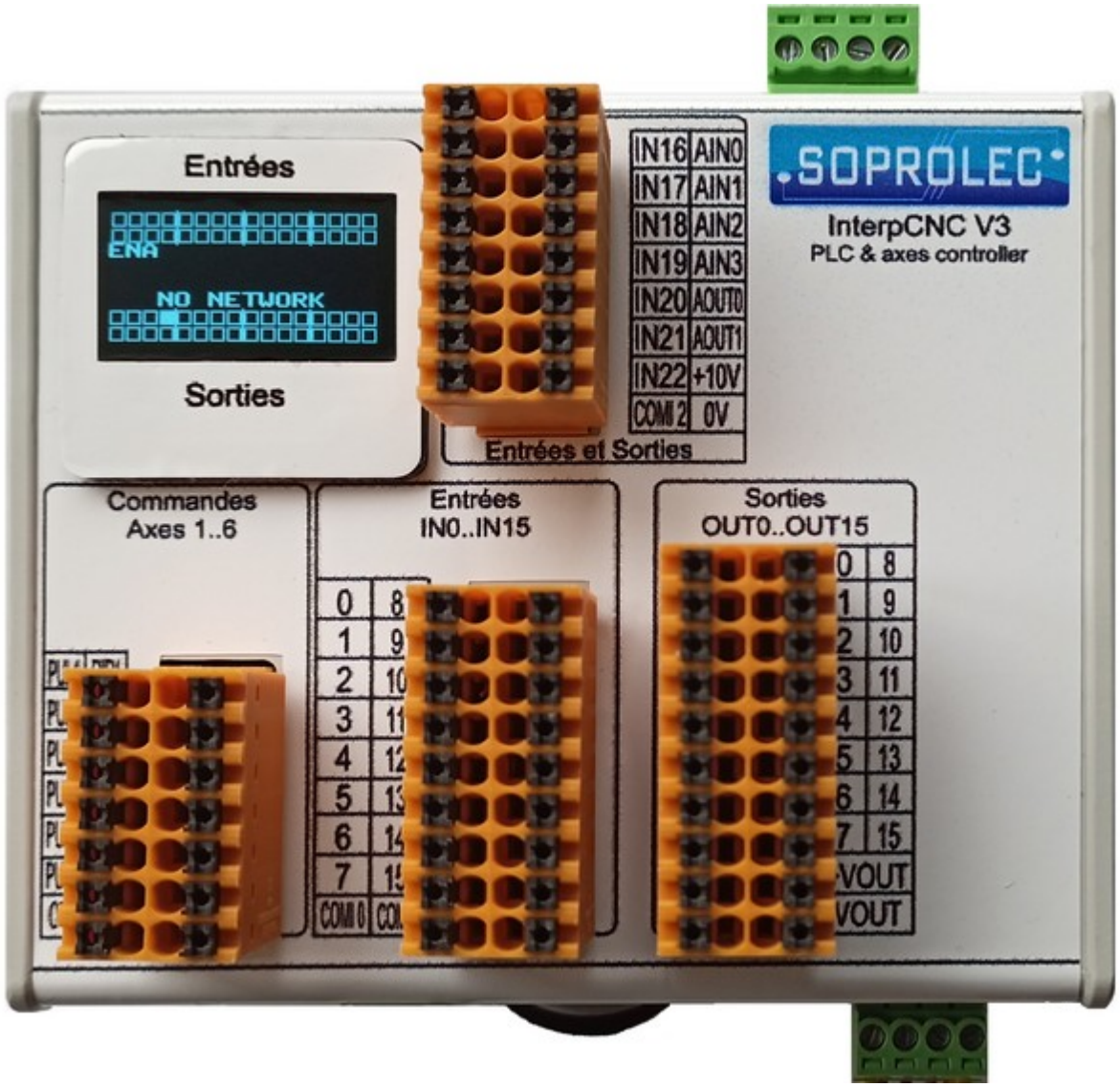
ICNCStudio Version : 1.0.0.83

- Ajout de système de blocs.
- Ajout de tableau de constantes.

Notice de l'InterpCNC V3

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

Carte d'axe
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 êtres 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).

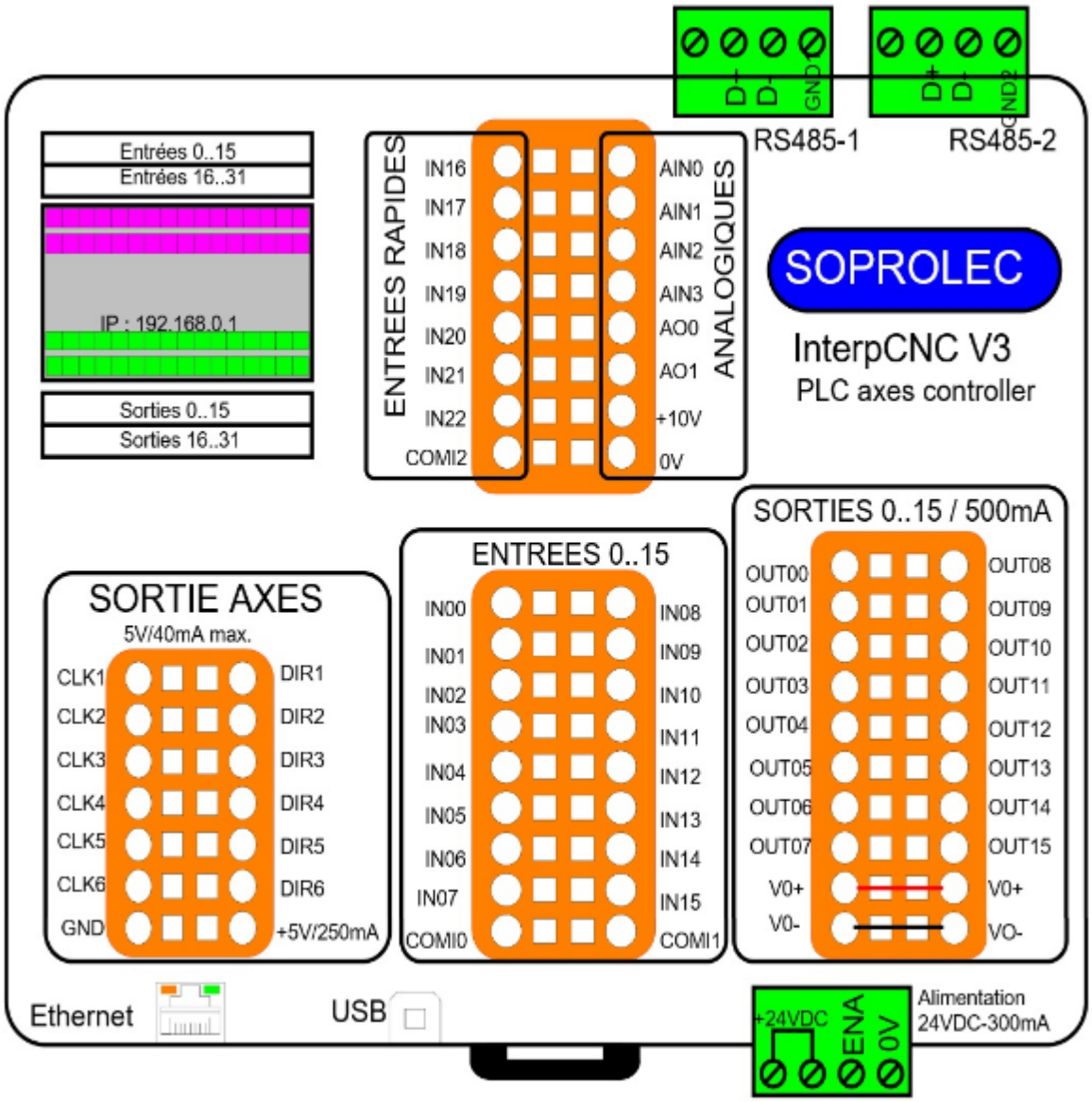
Trois interfaces de communication sont disponibles :

- 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-controleur (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, et 10, 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)

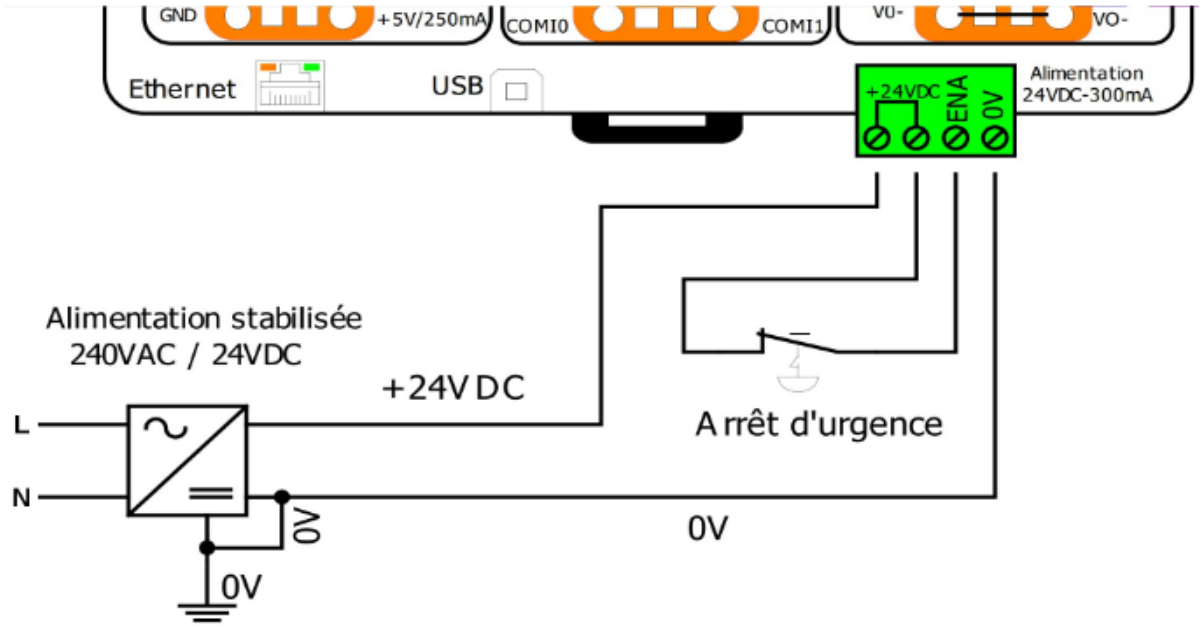
Quelle que soit le type d'interface utilisé, tous les échanges de données restent basés sur le protocole Modbus. La carte peut alors être configurée soit en Maître ou Esclave (voir tableau des paramètres).
En règle générale l' IHM occupe toujours la position maître, et la carte InterpCNC la

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

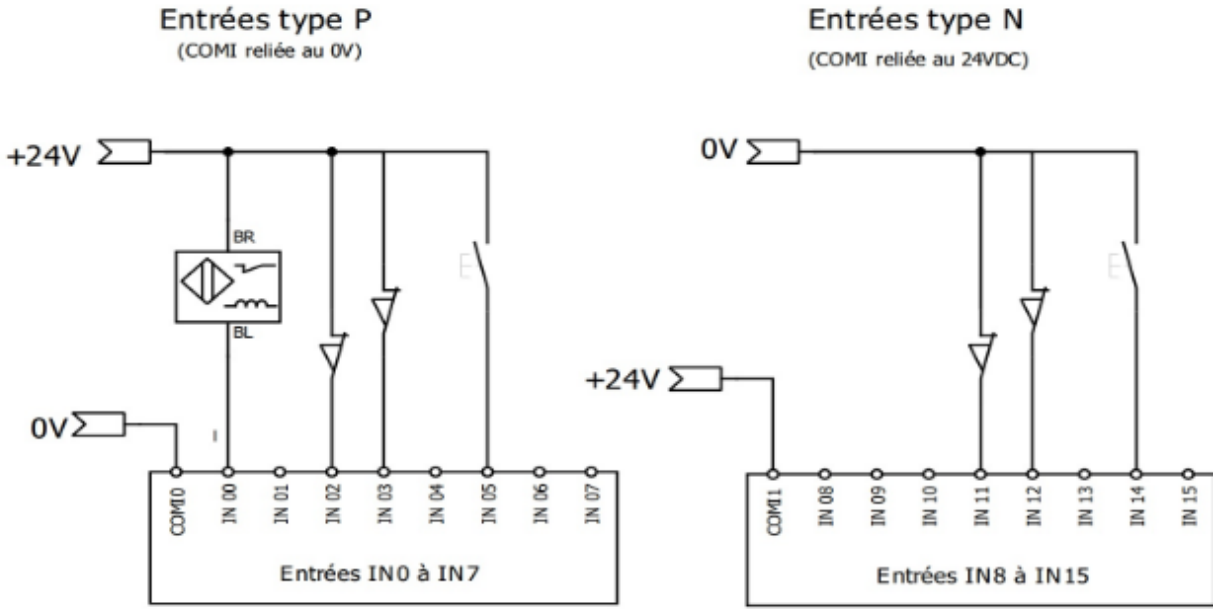
Alimentation, Arrêt d'urgence



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 (acquittement et reprise).

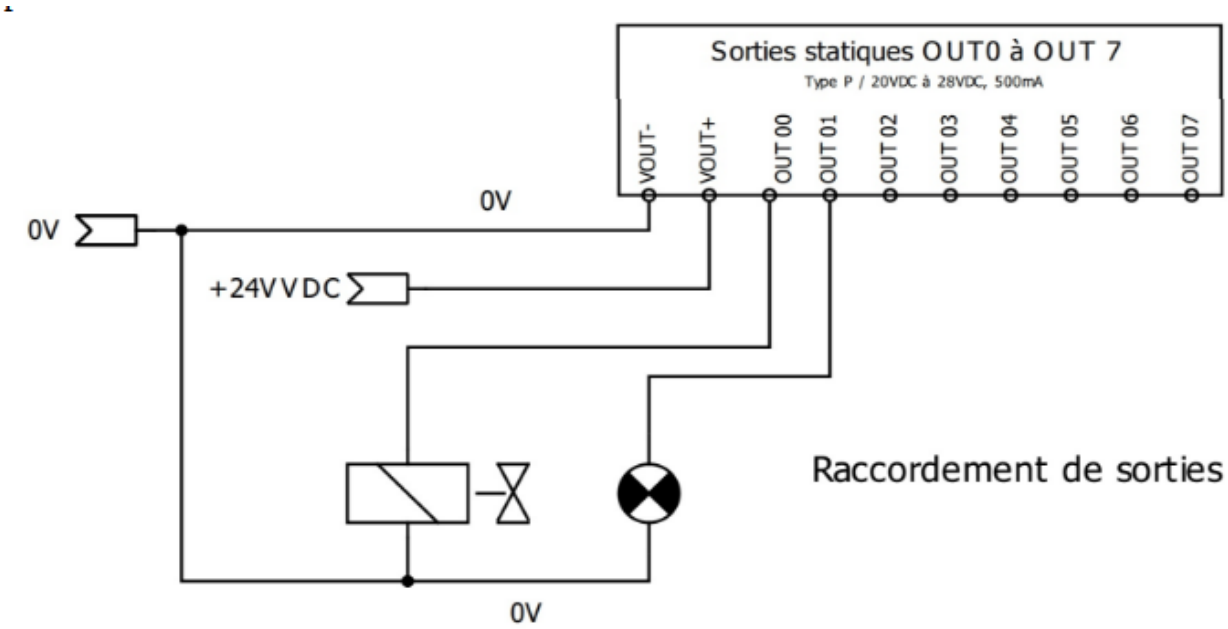
Raccordement des entrées

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

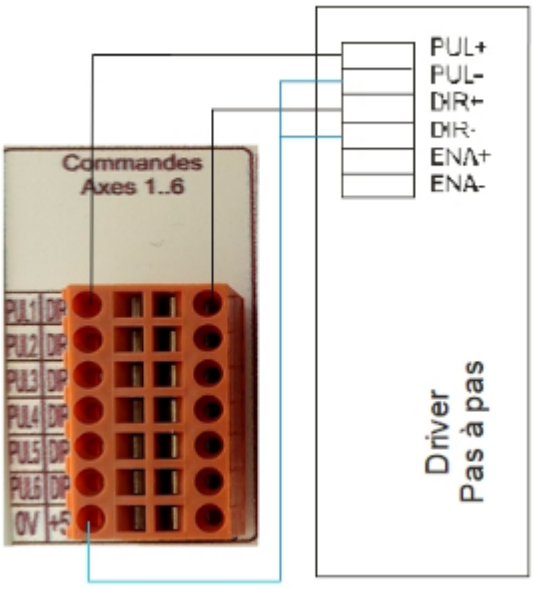
Les sorties statiques doivent être alimentées en externe (connecter le +24V à VOUT+, et le 0V à VOUT-). Elles sont de type PNP.
Exemple de raccordement :



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

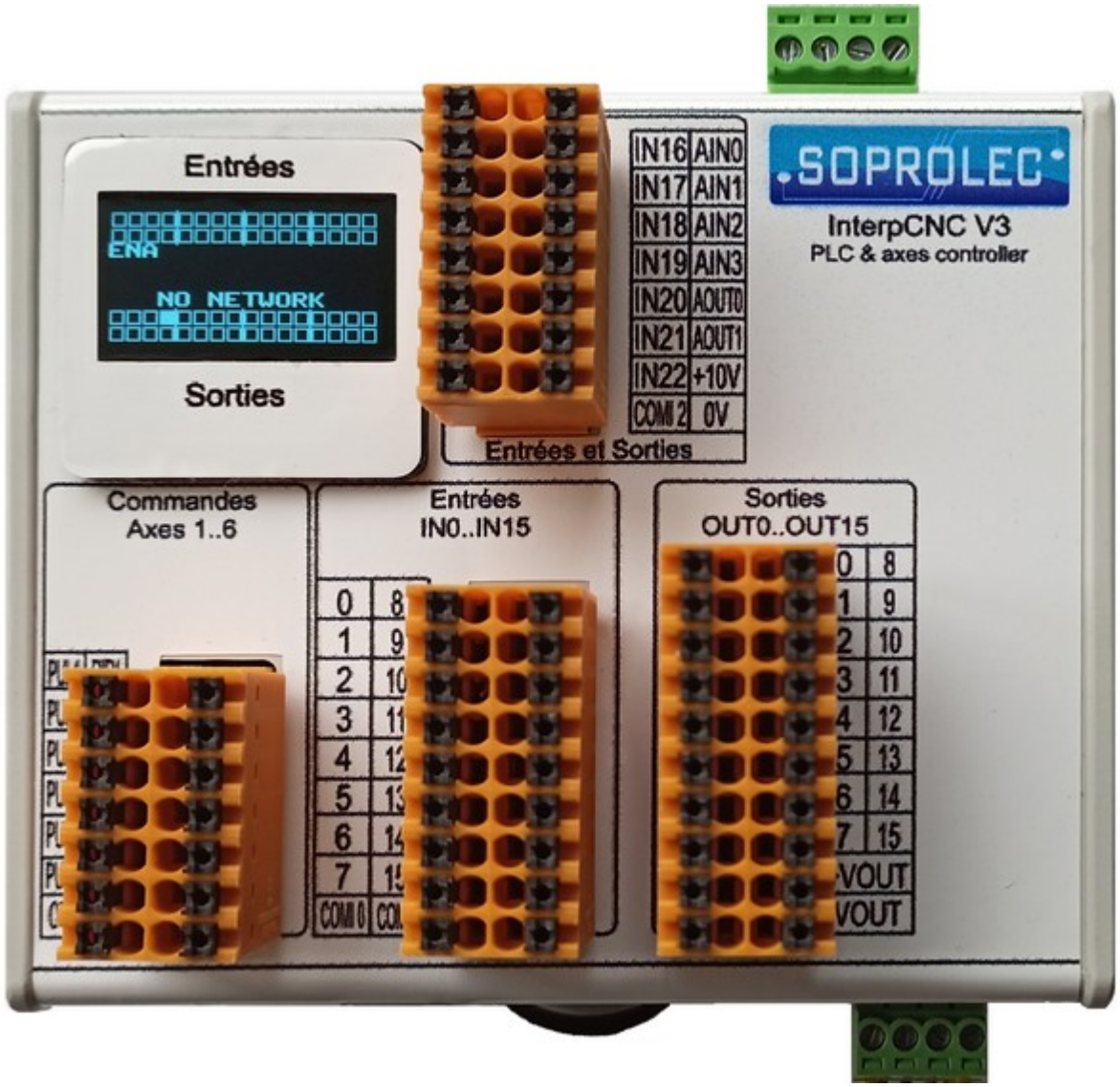
Les commandes pulses/direction de la carte sont à relier directement aux entrées correspondantes du driver.



Documentation_MODBUS

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

Carte d'axe
SOPROLEC
InterpCNC V3



Protocole de communication MODBUS

Introduction

Cette documentation est destinée à la programmation avancée de la carte InterpCNC 3.

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 3 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 êtres 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);

                        vid = startingAtVid.Substring(0, 4); // vid is four characters
long
                        string startingAtPid = deviceId.Substring(pidIndex + 4);

                        pid = startingAtPid.Substring(0, 4); // pid is four characters
long
                    }

                    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 peut se faire à l'aide de l'utilitaire ICNCStudio, mais également en Modbus:

Le tableau des paramètres et 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 faibles
264 à 271	Direction des mouvement (bit à 0 si mouvement négatifs, à 1 si mouvements 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 fonction RCP. A 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 non plus Modbus RTU. Commutation automatique sur le protocole Grbl 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 : Unde trame Linky a été reçue. Passe à 0 après lecture du bit avec 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	Status Divers 2 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 faible : 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 faible : Indicateurs d'axe en cours de séquence Probe 1023 Poids faible : Indicateurs d'erreur séquence de palpage (probe error)

'1026	Nombre de caractères dans le buffer de Print PLC Basic (commande ? Ou print)
'1027	Nombre de caractères dans le buffer de Trace PLC Basic (commande !)
1030 à 1041	Compteurs de position des axes AXE1 à AXE 6 en pulses (registres 32 bits 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 µs entre deux évènements 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 les 2 derniers fronts (montant ou descendant sans distinction). En mode compteur, ce sont uniquement les front montants qui sont pris en compte. Pour prendre en compte les front descendants, il faut inverser la polarité 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	Taile 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 (4096 maxi)
3001	Place disponible dans le buffer de communication CNC Grbl (1024 maxi)
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	THC Status : • b2 : THC autorisé. Activé par les commande THCOn ou THCOnAuto. Reset par commande THCStop. • b6 : THC Actif. Activé par THCOn, THCOnAuto ou THC Resume. Reset par THCStop ou THCPause. Indique que le THC est actif. • b9 : THC verrouillé pour cause de sous vitesse par rapport à la vitesse de découpe demandée • b11 : Temporisation avant activation en cours (paramètre Setting_THC_Delay) • b12 : Échantillonnage pour réglage auto tension THC en cours (si échantillonnage demandé, ce bit est à 1 aussi durant le délai initial) • b13 : Le résultat d'échantillonnage est en dehors de la tolérance. La valeur utilisée est bornée à la plage autorisée. Le bit repasse à 0 au stop THC
3013,,3014	(int32, mV) Mesure tension d'arc en mV. Mise à l'échelle automatique en fonction des paramètres d'offset et de tension max de mesure. Indication 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 non signés). L'entrée doit être configurée en mode Interruption ou compteur. En mode interruption, tous les changements d'état seront comptabilisés. En mode compteur, les fronts montants seront comptabilisés. Pour compter les fronts descendants, il faut donc inverser la polarité de l'entrée (voir paramètres 200).
2400 à 2411	Compteurs de position des axes AXE1 à AXE 6 (registres 32 bits signés)
2412 à 2423	Vitesses actuelles (en pulses/s) des axes en cours de déplacement (non 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 logiciel de pilotage externe ou par échantillonnage de la mesure.
2446	

	(int16_t, mV) Offset appliqué à la consigne tension d'arc en temps réelle pour ajustement manuel
2447	(uint16_t, %) : Override de vitesse sur déplacement THC (0% à 1000%) Ce registre agit au même titre que le paramètre de gain en vitesse des paramètres THC. Il est initialisé à 100 à la mise sous tension. Une valeur à 0 se traduira par un arrêt du THC. Il peut être modifié à tout moment avec prise en compte immédiate. La modification peut également se faire à l'aide d'une action d'écriture de registre bufferisée.
2800 à 2899	Indexes pour lectures/écritures indexées. Soit 50 registres 32 bits permettant 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 auto incrémenté 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 identifié 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**

Adresse	2000	2001
Paramètre	Commande ID	Axe ID
Valeur	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 »

Adresse	2000	2001
Paramètre	Commande ID	Axes bits
Valeur	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).

Adresse	2000	2001	2002	2003	2004	2005
Paramètre	Commande ID	Axe ID	Accel/Decel		Vitesse	
Valeur	102	1,,6	LW Accel	HW Accel	LW Fréquence	HW Fréquence

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	Commande 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 ou 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	2019	2020
Accélération		Décélération		Max Stroke		Tempo Reverse Direction	Home Position to set		Clearance	
LW Acceleration	HW Acceleration	LW Deceleration	HW Deceleration	LW Max Stroke	HW Max Stroke	Time in ms	LW Home Position	HW Home Position	LW Clearance	HW Clearance

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 palpage peuvent êtres lancées simultanément sur différents axes.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011
Param	Command ID	Axe ID	Input Number	Expected Input state	Accélération		Vitesse		Décélération		Max stroke (signed value → Direction)	
Valeur	107	1,,6	0,,255	0 or 1	LW Acceleration	HW Acceleration	LW Velocity	HW Velocity	LW Deceleration	HW Deceleration	LW Max Stroke	HW Max Stroke

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 ou le résultat du test lmogique est <> 0, Ces positions sont disponibles dans les Inputs registers 1330 à 1341.

NB : Plusieurs séquences de palpage peuvent êtres lancées simultanément sur différents axes.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
Param	Command ID	Axe ID	AND mask		XOR mask		Accélération		Vitesse		Décélération		Max stroke (signed value → Direction)	
Valeur	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 : palpage tant que l'entrée analogique est <= au seuil indiqué
Si opérateur = 1 : palpage tant que l'entrée analogique est >= au seuil indiqué

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 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 ou le seuil de déclenchement est atteint, Ces positions sont disponibles dans les Inputs registers 1330 à 1341.

NB : Plusieurs séquences de palpage peuvent êtres lancées simultanément sur différents axes.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
Param	Command ID	Axe ID	AIN number	Operator	Thresold mV	Accélération		Vitesse		Décélération		Max stroke (signed value → Direction)	
Valeur	109	1,,6	0,,7	0,,1	0,,10000	LW Acceleration	HW Acceleration	LW Velocity	HW Velocity	LW Deceleration	HW Deceleration	LW Max Stroke	HW Max 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_GREATHER_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 grather than 5000mV
// Accel = 10000Hs/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 reatch befon 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

Cette fonctionnalité permet de regrouper les informations qui vous intéressent dans une plage d'adresse Modbus continue. Il est ainsi possible d'optimiser le flux de communication.

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 ou 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 des 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 :

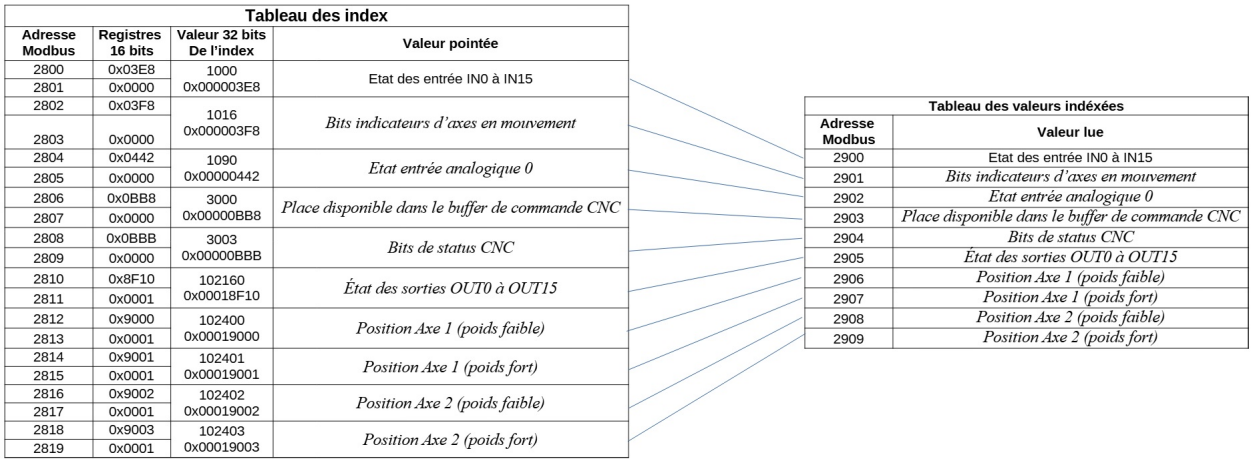


Tableau des index			
Adresse Modbus	Registres 16 bits	Valeur 32 bits De l'index	Valeur pointée
2800	0x03E8	1000 0x000003E8	Etat des entrée IN0 à IN15
2801	0x0000		
2802	0x03F8	1016 0x000003F8	*Bits indicateurs d'axes en mouvement*
2803	0x0000		
2804	0x0442	1090 0x00000442	*Etat entrée analogique 0*
2805	0x0000		
2806	0x0BB8	3000 0x00000BB8	*Place disponible dans le buffer de commande CNC*
2807	0x0000		
2808	0x0BBB	3003 0x00000BBB	*Bits de status CNC*
2809	0x0000		
2810	0x8F10	102160 0x00018F10	*État des sorties OUT0 à OUT15*
2811	0x0001		
2812	0x9000	102400 0x00019000	*Position Axe 1 (poids faible)*
2813	0x0001		
2814	0x9001	102401 0x00019001	*Position Axe 1 (poids fort)*
2815	0x0001		
2816	0x9002	102402 0x00019002	*Position Axe 2 (poids faible)*
2817	0x0001		
2818	0x9003	102403 0x00019003	*Position Axe 2 (poids fort)*
2819	0x0001		

Tableau des valeurs indéxées	
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 <=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(registres 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+n/2	
Paramètre	Commande ID	Longueur commande	Cmd		Cmd		Cmd		Cmd	
Value	1000	0,,255	Cmd[1]	Cmd[0]	Cmd[3]	Cmd[2]	---	---	Cmd[n]	Cmd[n-1]

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.

Adresse	2000	2001	2002
Paramètre	Commande ID	float	
Value	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.

Adresse	2000	2001	2002	2003	2004	2005	...
Paramètre	Commande ID	Indicateur	Float		Float		...
Value	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 registe 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 ID	Type d'action	N° sortie Analog.	Valeur
Value	1010	2	0 à 7	0 à 10000mV

Action de type 2 : écriture d'une sortie analogique

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Adresse registre	Valeur
Value	1010	3	2000 à 65535	0 à 65535

Action de type 3 : écriture d'un registre Modbus (Holding register)

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Not used	Not used
Value	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

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Durée mesure (ms)	Not used
Value	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

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Not used	Not used
Value	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)

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Not used	Not used
Value	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.

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Type d'action	Not used	Not used
Value	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.

Adresse	2000	2001
Paramètre	Commande ID	Durée temporisation en ms
Value	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 descendent 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 (HW)
Paramètre	Commande ID	Type d'action	Timeout (ms)	Action Timeout	Type attente	Seuil	Adresse modbus étendue	
Value	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 registe : 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
Paramètre	Commande ID	Override usinage en %	Override Hors matiere en %
Value	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.

Adresse	2000
Paramètre	Commande ID
Value	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.

Adresse	2000
Paramètre	Commande ID
Value	1102

Commande 1110 : Exécution d'une séquence de homing machine

Lancement d'une commande de homing suivant les paramètres internes pré-configurés.

Adresse	2000	2001
Paramètre	Commande ID	Axes concernés
Value	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 êtres 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 êtres donnés dans l'ordre croissant des indexes d'axe. Exemple, pour déplacer les axes X et Z, Indicateur vaudra 5, les les positions seront donnés 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 mouvement seront ainsi automatiquement limités.

Adresse	2000	2001	2002	2003	2004	2005	2006	2007	...
Paramètre	Commande ID	Indicateur	Float		Float		Float		...
Value	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.

Adresse	2000	2001
Paramètre	Commande ID	Code fonction
Value	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 de 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 donnée consistante, la lecture de l'Input Registre 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'idéale est donc de lire l'ensemble des informations requises dans ce laps de temps.

De nombreuse fonction 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.

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	Jour	Mois	Année
Value	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.

Adresse	2000	2001	2002	2003
Paramètre	Commande ID	heure	minutes	seconde
Value	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

Configurations

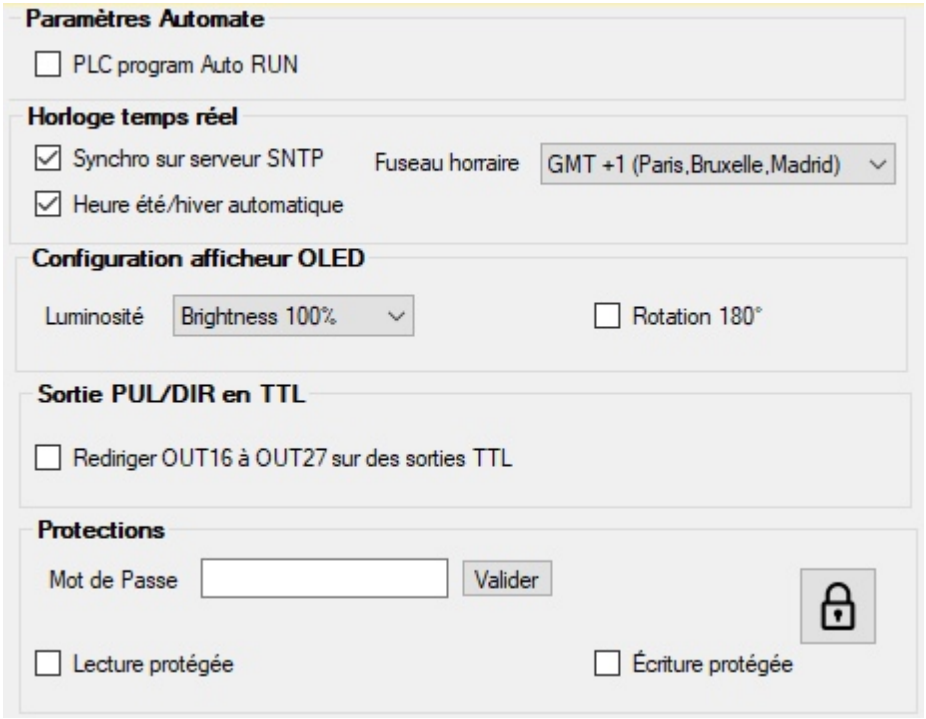
Ce chapitre traite de fonctionnalités avancées, offertes à la fois par le hardware et le firmware de la carte,
et configurables facilement depuis ICNCStudio.

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 précédement 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.

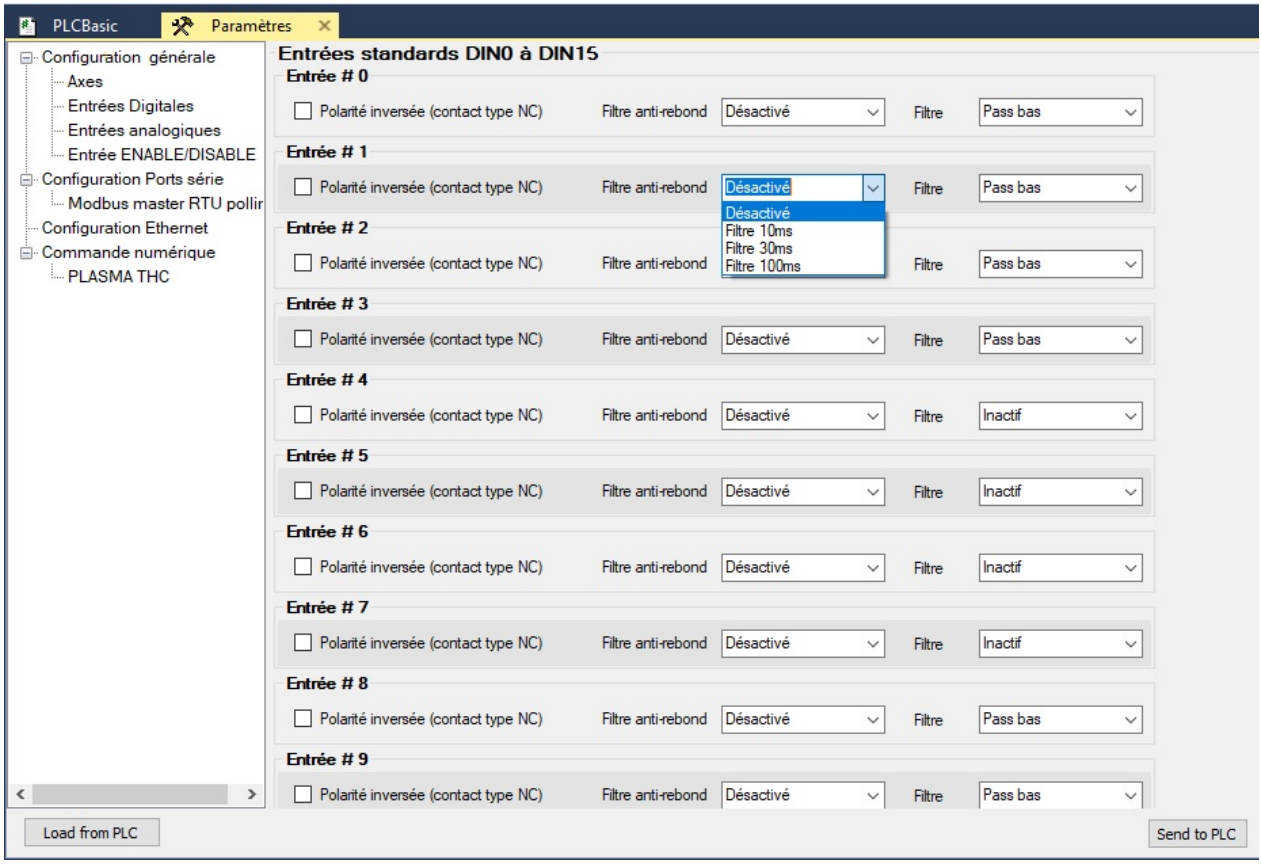
Astuce

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.

Entrees standards

Pour chaque entrée de DIN0 à DIN15, il est possible de d'inverser sa polarité.
Il est également possible de régler sa sensibilité, à l'aide de 3 niveaux de filtre Anti-Rebond (10, 30, ou 100ms).

Ce filtre peut utiliser le niveau de filtrage selectionné selon 3 modes:
Inactif, Passe bas, ou Echantillonnage.
Pour le choix du mode de l' Anti-Rebond voir l'explication donnée dans la rubrique **DIN** du tableau de paramètres.



Entrees rapides

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)

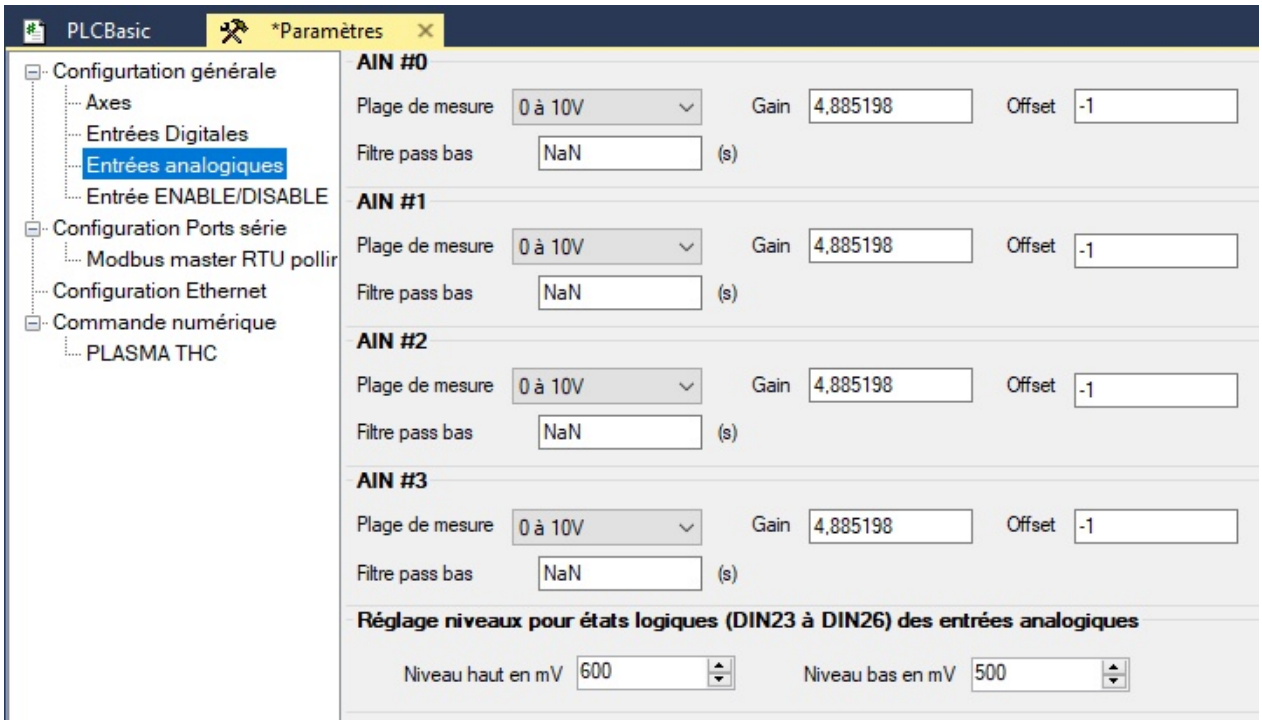
On peut donc disposer de 7 compteurs. Les paramètres 220 à 226 de la carte sont à configurer :
En mode 0 (standard), le rafraîchissement du tableau stockant l'état des entrées a lieu toutes les millisecondes.
En mode 1 (sur interruption « IT »), le rafraîchissement du tableau stockant l'état des entrées 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

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**".

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 rafraichissement 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

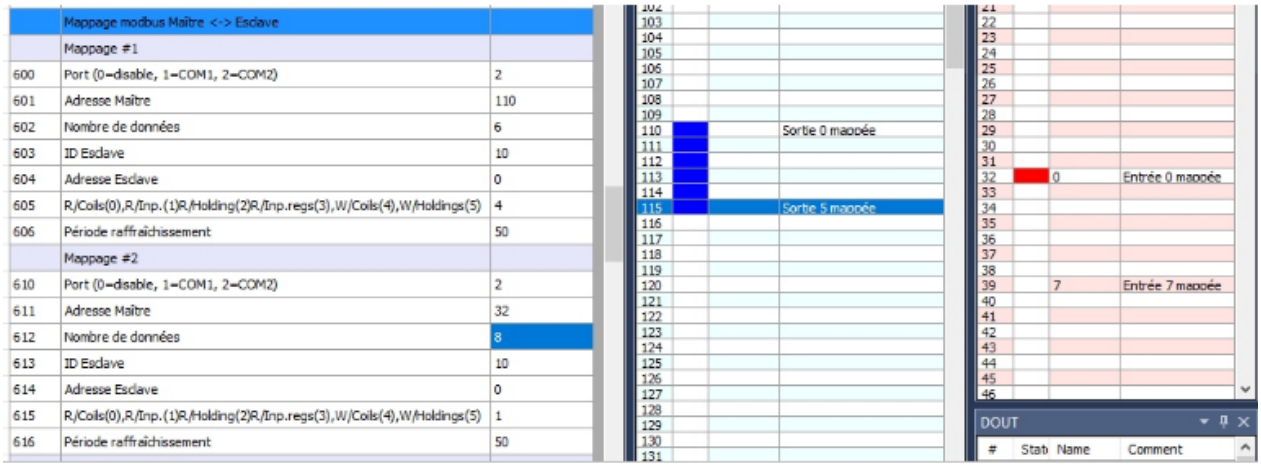
Raccordement Modbus RS485:

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 :



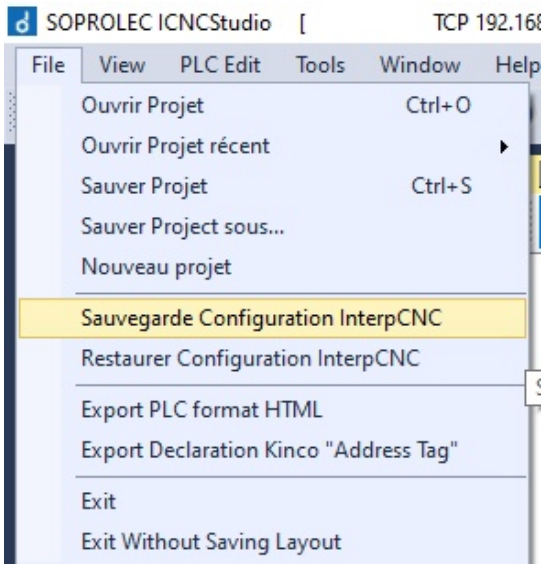
Ici, l'état des 6 sorties du module sera commandé par l'état des 6 Coils (bits Modbus) n° 110 à 115.
Les 8 entrées sont mappées sur les entrées virtuelles 32 à 39 (DIN).

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

A l'onglet **File** de la barre de Menu :



→ l'option « **Sauvegarde Configuration InterpCNC** ».
Celle-ci permet de sauvegarder sous la formes 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 des 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.