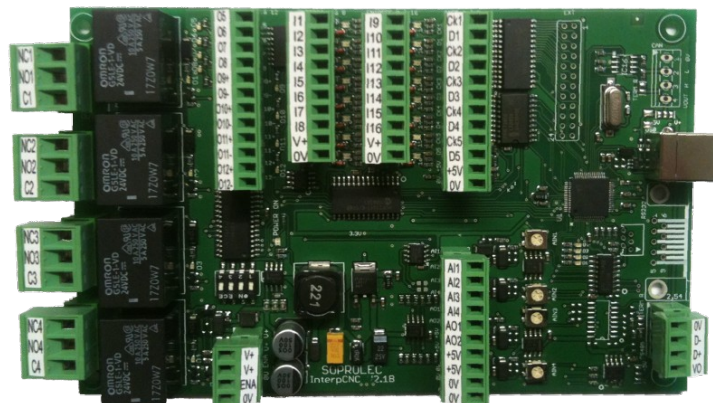


**SOPROLEC**  
ZAC DE L'EPINE  
72460 SAVIGNE L'EVEQUE  
Tél : +33 (0)2 4376 4476



## **Carte d'axe SOPROLEC InterpCNC V2.1**

**Interpréteur langage BASIC intégré**

## Table des matières

INTRODUCTION.....	3
GÉNÉRALITÉS.....	4
Gestion des interruptions.....	4
Interruptions périodiques.....	5
Interruptions Liées à l'état des entrées.....	5
Interruptions sur positions d' Axe.....	7
Gestion des erreurs.....	8
COMMANDES SPÉCIFIQUES.....	9
FONCTIONS SPÉCIFIQUES.....	18
Détails du registre de status (Status1 et Status2).....	25
Détails du registre de status étendu (Status 3 et Status4).....	26
CREATION D'UN GRAFCET AVEC L'INSTRUCTION « Select Case ».....	27
Exemple d'utilisation.....	27
UTILISATION DES CARTES D'AXES INTERPCNC V2 ET V2.1D EN MODE DMX.....	28
Introduction.....	28
I) Pré-requis.....	28
II) Raccordement.....	28
III) Utilisation du DMX dans votre programme.....	29
IV) Programme simple de démonstration.....	29
EXEMPLES DE PROGRAMMES.....	32
Sauvegarde périodique des positions en EEPROM.....	32
Activer/désactiver un clignotant sur front montant d'une entrée.....	32
Gestion de déplacement des axes par des entrées MARCHE et DIRECTION.....	33
Déplacements des axes X et Y par un joystick.....	35
Déplacements des axes X et Y par l'entrée codeur.....	39
Asservissement analogique d'un axe avec PID et entrée codeur.....	40
HISTORIQUE DES MODIFICATIONS.....	41

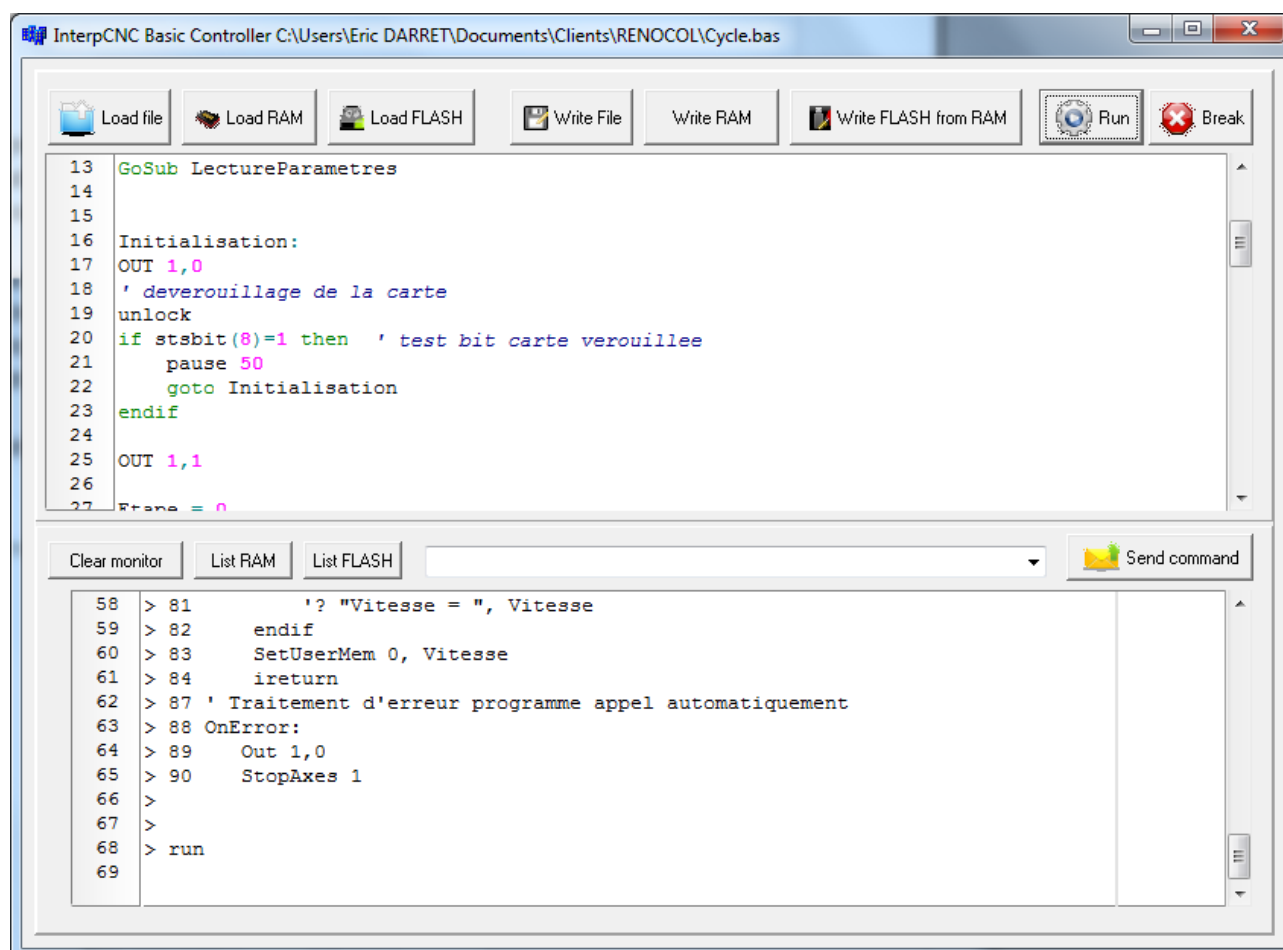
## INTRODUCTION

L'interpCNC V2.1 dispose d'un puissant interpréteur de langage BASIC intégré. Cet interpréteur permet de développer des applications d'automatisme autonomes ou en association avec une interface homme/machine communiquant avec la carte via le protocole Modbus.

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 TestCenter.



## GÉNÉRALITÉS

L'interpréteur Basic travail avec des variables qui peuvent êtres des chaînes de caractères ou des nombres réels.

Le temps de traitement moyen d'une instruction est de l'ordre de 15µs et peut varier en fonction de la charge du processeur.

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  $\pm 16777100$

### ***Gestion des interruptions***

Vous avez la possibilité de programmer deux types d'interruption.

- Les interruptions périodiques,
- Les interruptions liées à l'état des entrées.

Le temps de latence pour le traitement d'une interruption est  $< 100\mu s$ .

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

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

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

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 16

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 GetOut(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** InterruptionNo, EntreeNo, Type, Label

InterruptionNo : Numéro de l'interruption de 1 à 16

EntreeNo: Numéro de l'entrée à surveiller (1 à 32)

Type : Type de contrôle

1 pour contrôler tous les changements d'état

2 pour contrôler le passage de l'état 0 à l'état 1 de l'entrée

3 pour contrôler le passage de l'état 1 à l'état 0 de l'entrée

4 pour contrôler tous les changements d'état, une seule fois (ONESHOT)

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

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

Label : Label ou trouver le traitement d'interruption.

Exemple1 : 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 :*

*StopAxe 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 16

*AxeNumber* : 1 à 5 (sur Axes motorisés), 6 (sur Entrées codeurs),  
7 à 8 (sur Compteurs A et B sur entrées rapides)

*CapturePosition* : position cible (en pas moteur)

*Label* : Label de l'interruption

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

## ***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 Basic réalisera automatiquement un GoTo au label OnError si une erreur d'exécution se présente.

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

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

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

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

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

```

' Programme principal
Do
    ' Code de l'application
Loop

' Traitement d'erreur
OnError :
    StopAxes &H1F ' Arrêt de tous les axes
    OutAll 0 ' Mise à 0 des sorties

```

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

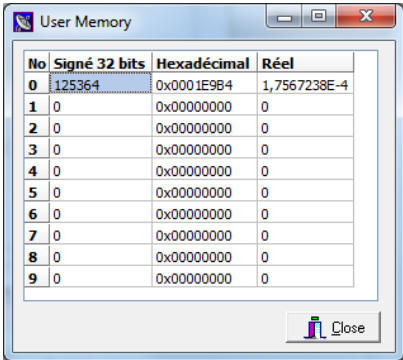


## COMMANDES SPÉCIFIQUES

Vous trouverez dans le tableau ci-dessous, les commandes qui ne sont pas en standard dans le langage Basic.

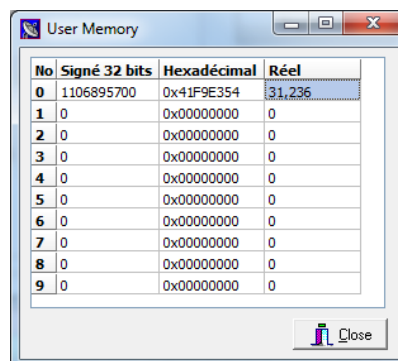
Elles permettent la gestions des axes et également, l'utilisation des différents périphériques de la carte InterpCNC.

<p><b>Pause</b> ppp</p> <p>ppp = durée en ms</p>	<p>Pause de ppp milliseconde dans l'exécution du programme. Les traitement d'interruption ne sont en revanche pas interrompus durant une pause. Cette fonction peut également être utilisé dans les traitements d'interruption.</p> <p>Exemple d'activation de la sortie 1 pendant une seconde: OUT 1,1 'Activation sortie 1 PAUSE 1000 'Pause 1000ms OUT 1,0 'Désactivation sortie 1</p>
<p><b>OUTTTL</b> Numéro, Valeur</p> <p>Numéro des sorties (Cartes 3 et 5 axes): <b>1</b> : CLK1, <b>2</b> : DIR1, <b>3</b> : CLK2, <b>4</b> : DIR2, <b>5</b> : CLK3, <b>6</b> : DIR3</p> <p>Numéro des sorties (Carte 5 axes uniquement): <b>7</b> : CLK4, <b>8</b> : DIR4, <b>9</b> : CLK5, <b>10</b> : DIR5</p> <p>Valeur : 0 ou 1</p> <p>NB : Il est préférable de n'utiliser comme sorties TTL, que celles des axes non utilisés. (Éventuellement la sortie DIR d'un axes utilisé peut être employée, à condition d'être sûr de tourner toujours dans le même sens).</p>	<p>Les sorties Pulses/Direction des axes peuvent être utilisées comme sorties TTL (0 ou 5V), ce qui peut être pratique dans le cas où vous avez besoin de plus de sorties physiques que celles normalement disponibles.</p> <p>Exemples: <b>OUTTTL</b> 2, 1 'Niveau haut sur sortie DIR1' <b>OUTTTL</b> 5, 0 'Niveau bas sur sortie CLK3'</p> <p>Il est possible d'inhiber l'effet des commandes d'axes (MoveAxe, Home, etc...) sur ces sorties pour ne pas interférer. Pour ce faire, dans le registre système 45, un bit est attribué à chaque sortie (Bit 0 pour CK1, Bit1 pour DIR1, etc...) → à 1, la sortie répond aux commandes d'axes et à la commande <b>OUTTTL</b> → à 0, les commandes d'axes sont inhibées, et la sortie ne répondra qu'à la commande <b>OUTTTL</b>.</p> <p>Ainsi: <b>SetSys</b> 45, 0 inhibe tous les mouvements d'axes sur toutes ces sorties (= <b>SetSys</b> 45, &amp;b00000000000000000) <b>SetSys</b> 45, 975 inhibe les mouvements d'axes pour CK3 et DIR3 (= <b>SetSys</b> 45, &amp;b00000001111001111)</p>
<p><b>OUT</b> Numéro, Valeur</p> <p>Numéro = 1 à 32 Valeur = 0 ou 1</p>	<p>Activation/désactivation d'une sortie tout ou rien (OUT 1 à OUT 32). Les sorties OUT 1 à OUT 12 sont des sorties physiques. Les sorties 13 à 32 sont des sorties virtuelles qui peuvent être utilisées pour la communication avec un dispositif externe (Modbus ou USB)</p>
<p><b>OUTALL</b> Valeur</p>	<p>Définition de l'état de toutes les sorties en même temps. Exemple : OUTALL &amp;H10 ' Activation sortie OUT5 OUTALL &amp;H12 ' Activation sortie OUT5 et OUT2</p>

<b>SetAna</b> Canal, Valeur Canal = 1 ou 2 (AO1 ou AO2) Valeur = 0 à 1023	Indique le niveau d'une sortie analogique en point convertisseur (0 à 1023). Exemple : SETANA 1, 512 ' Sortie analogique AO1 à 5V
<b>SetAnaV</b> Canal, Tension Canal = 1 ou 2 (AO1 ou AO2) Valeur = 0 à 10V	Indique le niveau d'une sortie analogique en Volt (0 à 10). Exemple : SETANAV 1, 5.51 ' Sortie analogique AO1 à 5,51V
<b>SetPrm</b> Numéro, Valeur Numéro : ID paramètre de 0 à 399	Écriture de la valeur d'un paramètre InterpCNC en connaissant son identifiant. Exemple : SETPRM 6, 1 ' Définition sens de rotation axe X
<b>SetPos</b> AxeNo, Valeur Axe : Numéro d'axe 1 à 5 Valeur : 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 Axe permet d'indiquer l'axe concerné par la commande. Exemple : SETPOS 3, 1000 ' Écrire 100 dans le compteur axe Y
<b>SetUserMem</b> MemNo, Valeur MemNo : Numéro mémoire de 0 à 9 Valeur : Valeur à écrire (entier)	Écriture dans la zone mémoire RAM utilisateur. Cette zone mémoire est accessible à l'aide du programme TestCenter (Menu Édition/Mémoire utilisateur) et permet par exemple l'affichage de variable durant la phase de mise au point du programme ou pour  l'échange de donnée entre application (Basic, Modbus ou USB)    Exemple : VariableX = 125364 SetUserMem 0, VariableX ' Copie la valeur de VariableX en mémoire

**SetUserMemF** MemNo, Valeur  
 MemNo : Numéro mémoire de 0 à 9  
 Valeur : Valeur à écrire (Réel)

Cette fonction est identique à SetUserMem mais permet d'écrire un nombre réel dans cette zone mémoire.  
 Cette zone mémoire est accessible à l'aide du programme TestCenter (Menu Édition/Mémoire utilisateur) et permet par exemple l'affichage de variable durant la phase de mise au point du programme ou pour l'échange de donnée entre application (Basic, Modbus ou USB)



No	Signé 32 bits	Hexadécimal	Réel
0	1106895700	0x41F9E354	31,236
1	0	0x00000000	0
2	0	0x00000000	0
3	0	0x00000000	0
4	0	0x00000000	0
5	0	0x00000000	0
6	0	0x00000000	0
7	0	0x00000000	0
8	0	0x00000000	0
9	0	0x00000000	0

Exemple :

PosXmm = GetPos(1) / 320.23

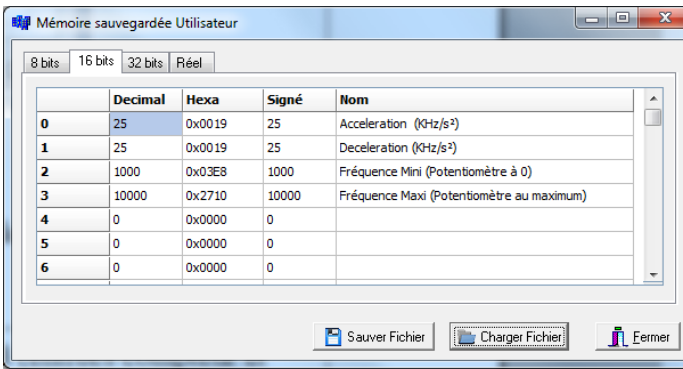
SetUserMemF 0, PosXmm ' Copie la position de l'axe X en mm X en mémoire

**SetCnt** CompteurNo, Valeur  
 Compteur No = 0, 1 ou 2  
 Valeur = Valeur du compteur

CompteurNo :

- 0 pour écrire la position codeur
- 1 pour écrire le compteur entrée A
- 2 pour écrire le compteur entrée B

Écriture dans les compteurs d'entrée rapide A et B ou dans le compteur codeur.  
 L'InterpCNC dispose de 2 entrées rapides qui peuvent être utilisées comme entrées de comptage.

<p><b>SetEEData8</b> Adresse, Valeur</p> <p>Adresse : 0 à 1023 Valeur : 0 à 255</p>	<p>Écriture dans la zone de mémoire Utilisateur sauvegardée. Vous disposez d'une zone de 1024 octets accessibles sous forme d'octet, d'entiers 16 bit, d'entiers 32 bits ou de réels. Cette zone mémoire est particulièrement adaptée à l'enregistrement de paramètres utilisateur et accessible également en Modbus ou par USB (Menu Édition/Éditeur EEPROM Utilisateur).</p> <p>Attention, cette zone mémoire est accessible sous différents formats mais il s'agit d'une zone unique. Donc, l'écriture de l'octet N°0 affectera l'entier 16 bit N°0, l'entier 32 bits N°0 et également le réel N°0,</p>  <p>Exemple : SetEEData8 0, 25</p>
<p><b>SetEEData16</b> Adresse, Valeur</p> <p>Adresse : 0 à 511 Valeur : 0 à 6553</p>	<p>Fonctionnement identique à la fonction SetEEData8 mais pour des entiers 16 bits</p>
<p><b>SetEEData32</b> Adresse, Valeur</p> <p>Adresse : 0 à 255 Valeur : 0 à <math>2^{32}-1</math></p>	<p>Fonctionnement identique à la fonction SetEEData8 mais pour des entiers 32 bits</p>
<p><b>SetEEDataFloat</b> Adresse, Valeur</p> <p>Adresse : 0 à 255 Valeur : <math>1.17549435e-38</math> à <math>3.40282347e+38</math></p>	<p>Fonctionnement identique à la fonction SetEEData8 mais pour des entiers 32 bits</p>

<p><b>SetMBit</b> Adresse, Valeur</p> <p>Adresse : Adresse Modbus des bits (256 à 512), Valeur : 0 ou 1</p>	<p>Écriture d'un bit dans la zone mémoire Modbus (bits en lecture/écriture nommés Coils) situés entre les adresses Modbus 256 et 512. Voir documentation Modbus InterpCNC pour plus de détails.</p> <p>La zone comprise entre les adresse 320 à 415 est disponible pour vos applications. Elle est particulièrement utile pour une communication entre l'application embarquée en Basic et un IHM en liaison Modbus.</p> <p>Exemple : SetMBit 256, 1 'Activation de la sortie OUT1</p>
<p><b>SetMReg</b> Adresse, Valeur</p> <p>Adresse : Registre Modbus (4096 à 5631) Valeur : Valeur 16 bits</p>	<p>Écriture d'un entier 16 bits dans la zone mémoire Modbus (registres en lecture/écriture nommés « Holding Registers ») situés entre les adresse Modbus 4096 et 5631. Voir documentation Modbus InterpCNC pour plus de détail.</p> <p>Exemple : SetMReg 4146, 512 'Mettre Sortie analogique AO1 à 5V</p>
<p><b>SRAdd</b> ArrayName, Valeur</p>	<p>Gestion d'un tableau en tant que registre à décalage. La fonction SRAdd décale toutes les valeurs d'un tableau déclaré par DIM(x) et écrit Valeur à l'index 0 du tableau. Exemple : DIM tab(3) tab(0)=0 tab(1)=1 tab(2)=2 SRAdd tab, 5 Le tableau tab contient alors les valeurs : 5 pour tab(0), 0 pour tab(1) et 1 pour tab(2)</p>
<p><b>SRFill</b> ArrayName, Valeur</p>	<p>Initialisation de l'ensemble des valeur d'un tableau déclaré par DIM(x) avec la valeur Valeur.</p>
<p><b>Unlock</b></p>	<p>Déverrouillage de la carte. Lorsque la carte est verrouillée, les commandes de déplacement ou d'activation des sorties sont inactives.</p>
<p><b>Lock</b></p>	<p>Verrouillage de la carte. Lorsque la carte est verrouillée, les commandes de déplacement ou d'activation des sorties sont inactives.</p>
<p><b>MoveAxe</b> Axe, Accel, Vitesse, Decel, Position</p> <p>Accel : Accélération en Khz/s Vitesse : Fréquence en Hz des pulses Decel : Décélération en Khz/s Position : Cible en pas moteur</p>	<p>Déplacement d'un axe à la position indiquée. L'envoi d'une nouvelle commande MoveAxe sur un axe en cours de déplacement aura pour effet de modifier la vitesse et/ou la cible.</p> <p>Cette commande permet de gérer des déplacements indépendants des axes. La vitesse suit un profil trapézoïdale. La fréquence Start/Stop utilisée est celle du paramètre InterpCNC N°</p> <p>Exemple : MoveAxe 1, 25,10000,25,50936 'Déplacement X MoveAxe 5, 25,80000,25,22541'Déplacement B</p>

<p><b>MoveInterp</b> AxesBit, Vitesse, Targets...</p> <p>AxesBit : Indique les axes à déplacer          Bit 0 = 1 pour déplacer l'axe X          Bit 1 = 1 pour déplacer l'axe Y          Bit 2 = 1 pour déplacer l'axe Z          Bit 3 = 1 pour déplacer l'axe A          Bit 4 = 1 pour déplacer l'axe B</p> <p>Vitesse : Fréquence en Hz des pulses sur l'axe majeur          Targets : Cibles des axes à déplacer</p>	<p>Déplacement interpolé de 2 axes ou plus,          La Vitesse indiquée en Hz est celle de l'axe majeur (celui qui a le plus grand déplacement à réaliser).          Les vitesses des autres axes seront calculées pour que les axes arrivent en même temps aux cibles indiquées.</p> <p>La fréquence Start/Stop utilisée est celle du paramètre InterpCNC N° 5          L'accélération/décélération utilisée est celle du paramètres InterpCNC N° 6</p> <p>L'envoi de plusieurs commande de déplacement interpolés est possible. Une seule sera traité à la fois. Les commandes suivantes sont placées dans un buffer d'exécution et traitées les unes à la suite des autres.</p> <p>Exemple :          MoveInterp 3, 10000, 43567, 21237 ' Move X and Y          MoveInterp &amp;H18, 5000, 12456, 14325 ' Move A and B</p>
--	--

<p><b>Probe</b> AxeNo, Direction, InNo, InValue, MaxStroke, Accel, Speed, Decel</p> <p>AxeNo : Axe à déplacer (1 à 5)  Direction : 1 si positif, -1 si négatif  InNo : Numéro de l'entrée de détection  InValue : Niveau attendu (0 ou 1) de l'entrée  MaxStroke : Limite de course de recherche (en pas)  Accel : Accélération (Khz/s<sup>2</sup>)  Speed: Vitesse déplacement (en Hz)  Decel : Décélération (Khz/s<sup>2</sup>)</p>	<p>Déplacement d'un axe jusqu'à la détection d'une entrée.  Lors de l'appel à cette fonction, la carte InterpCNC prend en charge le déplacement de l'axe indiqué. Ce déplacement est automatiquement arrêté en cas d'erreur ou au moment où l'entrée souhaitée passe dans l'état indiqué.  La position de l'axe correspond alors à la position d'enclenchement du capteur +/- la distance de décélération.  La position exacte de détection peut être obtenue avec la commande GetSys(20)</p> <p>Il s'agit d'une fonction non bloquante. Il vous appartient donc de traiter la fin d'exécution de la fonction à l'aide des bits de status « B12 : Probe In Progress » et « B13 : Probe Error »  Durant le mouvement, le bit « Probe In Progress » est actif et donc StsBit(12) retourne 1.  Si un des paramètres d'appel à la fonction Probe est erroné (par exemple, numéro d'entrée invalide, Direction différente de 1 ou -1, Accel, Vitesse ou Decel nul...), la fonction est ignorée et le bit « Probe Error » est activé. Ce bit d'erreur est automatiquement remis à 0 lors d'un nouvel appel à la fonction.  Le bit d'erreur sera également positionné si l'état attendu de l'entrée du capteur n'arrive pas dans la course maximale indiquée par MaxStroke.  Si l'entrée attendue est déjà active, il n'y aura pas de déplacement et pas d'erreur.</p> <p>Exemple 1:  Probe 1, -1, 9, 0, 10000, 50, 5000, 100</p> <p>Déplacement de l'axe X en sens négatif jusqu'à ce que l'entrée 9 passe à 0 (contact de type NC), sur une distance maximale de 10000 pas. Le mouvement se faisant à la vitesse de 5000Hz, avec une accélération de 50 Khz/s<sup>2</sup> et une décélération de 100KHz/s<sup>2</sup></p> <p>Exemple 2 :</p> <p>Probe 3, 1, 11, 0, 10000, 50, 5000, 100  'Attendre fin de Probe ou Erreur  DO WHILE (StsBit(12)=1 AND StsBit(13)=0)  'Traitement en attente de fin de recherche de capteur...  LOOP  if StsBit(13)=1 then  'Traitement de l'erreur  ? "Erreur palpée"  else  ? "Palpée terminé avec succès"  ? "Position capteur = ", GetSys(20)  endif</p> <p>Déplacement de l'axe 3 en Positif jusqu'au passage à 0 de l'entrée 11 et affichage d'un message en fin de traitement,</p>
---	---

<p><b>Home</b> AxeNo, InNo, InValue, accel, HighSpeed, decel, course, lowSpeed, posHoming, [Simulation]</p> <p>AxeNo : Axe à initialiser (1 à 5)  InNo: Numéro de l'entrée utilisée pour le référencement,  InValue : Etat de l'entrée lorsque le contact est activé,  Accel : accélération mouvement rapide vers le capteur  HighSpeed : Vitesse rapide vers le capteur,  Decel : décélération mouvement rapide vers le capteur,  LowSpeed : vitesse dégagement du capteur,  PosHoming : Position initiale appliqué au compteur de position en fin de cycle,</p> <p>[Simulation] : paramètre optionnel à 1 durant la mise au point.</p>	<p>Lancement d'un cycle de référencement sur un axe.  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.</p> <p>Le cycle se déroule en 3 étapes :</p> <ul style="list-style-type: none"> <li>• Déplacement de l'axe dans le sens négatif jusqu'à la détection d'un front sur l'entrée indiquée.</li> <li>• Lors de la détection du front, arrêt avec rampe de l'axe puis, déplacement lent dans le sens positif jusqu'à la détection d'un nouveau front opposé au précédent.</li> <li>• Lors de la détection de ce nouveau front (qui correspond au dégagement du capteur), arrêt de l'axe puis initialisation de la position à la valeur indiqué dans les paramètres de la fonction.</li> </ul> <p>Pour chaque axes, deux bits de status sont associés à cette commande. Le premier indique que le homing de l'axe en question est en cours. Le second, permet de déterminer si le cycle s'est déroulé correctement (voir bits de status 50 à 59). Le bit d'erreur est automatiquement remis à 0 lors d'un nouvel appel à la fonction Home pour l'axe en question.</p> <p>Durant la mise au point du programme, il est possible d'ajouter le paramètre simulation à 1. Cela a pour effet de valider la séquence de prise d'origine sans erreur et sans mouvement.</p> <p>Exemple d'utilisation :</p> <pre>' Origine axe 1 sur entrée N°1 type NC sur une course maxi de 500000 pas. Position d'origine initialisée à 0 Home 1, 1, 0, 25, 10000, 25, 500000, 1000, 0 ' Origine axe 2 sur entrée N°2 type NC sur une course maxi de 1000000 pas. Position d'origine initialisée à 0 Home 2, 2, 0, 25, 20000, 25, 1000000, 1000, 0</pre> <pre>DO WHILE (StsBit(50) OR StsBit(52))   'Traitement en attente de fin de prise d'origine axe 1 et 2 LOOP  if StsBit(51)=1 then   'Traitement de l'erreur origine axe 1   ? "Erreur Homing axe 1" if StsBit(53)=1 then   'Traitement de l'erreur origine axe 2   ? "Erreur Homing axe 2" endif</pre>
--	---



<p><b>SetTick</b> InterruptNo, Periode, Label</p> <p>InterruptNo : Numéro d'interruption (1 à 20)          Periode : période des interruption en ms          Label : Label de la routine de traitement.</p>	<p>Permet de définir une interruption périodique.</p>
<p><b>SetInputInt</b> InterruptNo, Type, InNo, Label</p> <p>Définir une traitement d'interruption lors d'un événement sur l'une des entrées.          Type = 1 à 6</p>	<p>Le type permet d'indiquer le moment ou l'interruption est générée.          Type = 1 =&gt; Interruption sur changement d'état de l'entrée,          Type = 2 =&gt; Interruption si l'entrée passe de 0 à 1          Type = 3 =&gt; Interruption si l'entrée pas de 1 à 0          Type = 4 =&gt; 4 pour contrôler tous les changements d'état, une seule fois (ONESHOT)          Type = 5 =&gt; pour contrôler le passage de l'état 0 à l'état 1 de l'entrée, une seule fois (ONESHOT)          Type = 6 =&gt; pour contrôler le passage de l'état 1 à l'état 0 de l'entrée, une seule fois (ONESHOT)</p> <p>Exemple :          SetInputInt 2, 1, 8, OnIn9</p>
<p><b>SetTimer</b> NomVariable, Durée_ms</p>	<p>Initialisation d'une variable de type timer. La durée est exprimée en milliseconde. Commande à utiliser avec la fonction GetTimer</p> <p>Exemple :          SetTimer Tempo1, 500</p>

## FONCTIONS SPÉCIFIQUES

Les fonctions se distinguent des commandes par le fait qu'elles retournent une réponse.

<b>In</b> (NuméroInput) InputNo : Numéro entrée 1 à 32  Résultat = 0 ou 1	Lecture de l'état d'une entrée.  Exemple :  if In(1)=1 then ... endif
<b>Ain</b> ( CanalNo ) CanalNo : Numéro entrée analogique 1 à 4  Résultat = 0 à 1023 points	Lecture de l'état d'une entrée analogique en points.  Exemple : Activation de OUT 1 si entrée analogique >511 If Ain(1) > 511 then Out 1, 1 else Out 1, 0 endif
<b>AinV</b> ( CanalNo ) CanalNo : Numéro entrée analogique 1 à 4  Résultat = 0 à 10.0 volts	Lecture de l'état d'une entrée analogique en Volt.  Exemple : Activation de OUT 1 si AI1 > 3.30V  If Ain(1) > 3.30 then Out 1, 1 else Out 1, 0 endif
<b>GetUserMem</b> ( MemoryNo ) MemoryNo : Numéro de la mémoire de 0 à 9	Lecture de la valeur présente en mémoire utilisateur. La lecture considère que la valeur est un entier 32 bits.  Voir SetUserMem pour plus de détail
<b>GetUserMemF</b> ( MemoryNo ) MemoryNo : Numéro de la mémoire de 0 à 9	Lecture de la valeur présente en mémoire utilisateur. La lecture considère que la valeur est nombre réel.  Voir SetUserMemF pour plus de détail
<b>GetEEData8</b> ( Adresse )  Adresse : 0 à 1023	Lecture d'un octet 8 bits dans la zone mémoire sauvegardée.  Exemple :  Valeur = GetEEData8(12)
<b>GetEEData16</b> ( Adresse )  Adresse : 0 à 511	Lecture d'un mot 16 bits dans la zone mémoire sauvegardée.  Exemple :  Valeur = GetEEData16(3)

<b>GetEEData32( Adresse )</b>  Adresse : 0 à 255	Lecture d'un mot 32 bits dans la zone mémoire sauvegardée.  Exemple :  Valeur = GetEEData32(9)
<b>GetEEDataFloat( Adresse )</b>  Adresse : 0 à 255	Lecture d'un nombre réel dans la zone mémoire sauvegardée.  Exemple :  Valeur = GetEEDataFloat(9)
<b>GetPos( AxeNo )</b>  AxeNo : Numéro 1 à 5 de l'axe  Résultat : position de l'axe	Lecture du compteur de position d'un axe. Le retour est un entier 32 bits signé.
<b>GetPrm( Numéro)</b> Numéro : ID paramètre de 0 à 399  Résultat : Valeur du paramètre (entier signé)	Lecture de la valeur d'un paramètre InterpCNC en connaissant son identifiant. Exemple : NoEntree = GetPrm (104) ' Lecture du numéro d'entrée fin de course X-
<b>GetSys(VariableNo)</b>  VariableNo : index des variables système  Lecture de variables internes à la carte InterpCNC	Certaines variables internes au fonctionnement de la carte peuvent être obtenues à l'aide de la fonction GetSys(VariableNo), Les valeurs de VariableNo peuvent être : 11 à 15 pour lire la fréquence actuelle de déplacement des axes 1 à 5 20 : La position résultante du dernier appel à la fonction Probe, 100 : L'état du clavier matriciel connecté sur le connecteur d'extension, 200 : Le nombre d'octets reçus sur la liaison Modbus 201 : Le nombre d'octets transmis sur la liaison Modbus 202 : Le numéro de la dernière commande Modbus reçue, 203 : Le nombre de commande Modbus Broadcast reçues, 204 : Le nombre de trames Modbus reçues,
<b>GetMBBit( Numéro bit Modbus)</b>	Lecture d'un bit dans l'espace d'adressage Modbus.  Exemple :  GetMBBit(320)  Retourne l'état du premier bit utilisateur Modbus
<b>GetInAll</b>	Lecture de l'état des entrées.  Exemple :  Entree = GetInAll if (Entree and &H7) = &H7 then ' Code si les entrées 1 à 4 sont actives ... Endif

<b>GetOutAll</b>	Lecture de l'état actuel des sorties
<b>GetDIPSwitch</b>	Retourne l'état des 4 DIP Switch présents sur la carte.
<b>GetOut( SortieNo )</b> SortieNo = Numéro 1 à 32 de la sortie	Lecture de l'état actuel d'une sortie.  Exemple : ' Si la sortie 1 est active, la remettre à 0 if GetOut(1)=1 then Out 1, 0
<b>GetEncoder</b>  Résultat : entier signé	Lecture du compteur associé aux entrées codeur A et B.  Exemple :  PositionCodeur = GetEncoder Print PositionCodeur
<b>GetCnt( CompteurNo )</b> CompteurNo = Numéro de l'entrée compteur 1 ou 2  Résultat : Entier non signé	Lecture d'un des compteurs associés aux entrées rapides A et B. Ces compteurs sont utilisables si le paramètre « Entrée codeur » est configuré en mode Compteur.
<b>DFM( EntreeNo )</b> EntreeNo : Numéro de l'entrée 1 à 32  Résultat : 0 ou 1	Détection d'un front montant sur une entrée IN1 à IN32  Cette fonction permet de détecter le passage de l'état 0 à l'état 1 entre deux appels à cette fonction.  Exemple : ' Si front montant sur IN1, lancer le cycle if DFM(1)=1 then goto StartCycle
<b>DFD( EntreeNo )</b> EntreeNo : Numéro de l'entrée 1 à 32  Résultat : 0 ou 1	Détection d'un front descendant sur une entrée IN1 à IN32  Cette fonction permet de détecter le passage de l'état 1 à l'état 0 entre deux appels à cette fonction.  Exemple : ' Si entrée 3 passe à 0, traitement fin de course if DFD(3)=1 then goto FinDeCourse

<p><b>DFMBit</b>( NoFront, Etat)  NoFront : N° de variables de détection de front 1 à 32  Etat : Variable dont on souhaite surveiller l'état.</p>	<p>Fonction similaire dans le principe à la fonction DFM mais applicable à une variable quelconque.  Vous disposez de 32 cellules de détection de front.  La fonction retourne 1 quand la variable testée passe de la valeur 0 à une valeur autre.  La variable testée peut donc être le résultat d'un test ou une variable.</p> <p>Exemple 1:</p> <p>On souhaite réaliser une action lorsque la position de l'axe X passe au delà de 10000. La méthode classique impose de mémoriser la position actuelle de X et de la comparer à la nouvelle position :</p> <pre> MemoirePosX = GetPos(1) do   if (GetPos(1) &gt; 10000) and (MemoirePosX&lt;=1000)) then     ' Mémorisation nouvelle position     MemoirePosX = GetPos(1)     'Code exécuté si position X passe au delà de 10000   endif loop </pre> <p>Ce code peut avantageusement être remplacé par le code suivant (plus rapide et plus compact) :</p> <pre> if DFMBit( 1, GetPos(1)&gt;10000) then   'Code exécuté lorsque l'axe X passe au delà de 10000 points   ... endif </pre> <p>Exemple 2 :</p> <p>On souhaite réaliser une action lorsque la sortie OUT1 passe de l'état 1 à l'état 0.</p> <pre> if (DFMBit(2, GetOut(1)=0) then   ' Code exécuté lorsque la sortie OUT1 passe de 1 à 0 endif </pre> <p>Exemple 3 :</p> <pre> if DFMBit(3, AIN(1)) then   ' Code exécuté lorsque l'entrée analogique AI1 devient &gt; 0 endif </pre>
<p><b>DFDBit</b>( NoFront, Etat)  NoFront : N° de variables de détection de front 1 à 32  Etat : Variable dont on souhaite surveiller l'état.</p>	<p>Fonctionnement identique à DFMBit mais pour la détection du passage à 0 d'une variable.</p>

<p><b>IsMBReceived</b> Retourne 1 si une trame Modbus a été reçue.</p>	<p>IsMBReceived = 1 indique qu'une trame Modbus a été réceptionnée par la carte. Cette fonction est équivalente à la commande : StsBit(49) L'appel à cette fonction provoque la remise à 0 de ce bit de status correspondant.</p> <p>Exemple : Gestion d'un timeout de réception Modbus. Lors de la réception d'une trame, on réarme le timer d'une interruption périodique.</p> <p>'Si réception Modbus, réarmer Timer 250ms if IsMBReceived=1 then SetTick 1, 250, OnMBTimeOut</p> <p>'Interruption traitement timeout OnMBTimeOut :   'Traitement timeout réception ModBus   ...   iReturn</p>
<p><b>IsMBPrmChanged</b> Retourne 1 si le contenu de la mémoire utilisateur EEPROM a été modifié.</p>	<p>IsEEDataChanged = 1 indique que le contenu de la mémoire utilisateur EEData a été modifié soit par une commande Modbus, soit via la liaison USB (menu Edition/Editeur EEPROM Utilisateur).</p> <p>Cette fonction retourne l'état du bit de status StsBit(50). L'appel à cette fonction provoque la remise à 0 du bit de status correspondant.</p> <p>Exemple :</p> <p>Des paramètres sont stockés en EEPROM. On souhaite prendre en compte en dynamique le changement de paramètres.</p> <p>If IsMBPrmChanged=1 then GoSub LoadUserParameters ... suite du code</p> <p>' Subroutine de chargement des paramètres application LoadUserParameters :   ' Chargement paramètres applications   ResolutionX = GetEEDataFloat(0)   ResolutionY = GetEEDataFloat(1)   Vitesse = GetEEDataFloat(2)   Return</p>
<p><b>Max( Valeur1, Valeur2)</b></p>	<p>Retourne la valeur maximale entre les valeurs Valeur1 et Valeur 2 <u>Attention</u> : Valeur 1 et Valeur 2 sont traités comme des entiers.</p>
<p><b>Min( Valeur1, Valeur2)</b></p>	<p>Retourne la valeur minimale entre les valeurs Valeur1 et Valeur 2 <u>Attention</u> : Valeur 1 et Valeur 2 sont traités comme des entiers.</p>
<p><b>Limite( Valeur, Mini, Maxi)</b></p>	<p>Retourne Valeur bornée entre Mini et Maxi</p>

<b>Timer</b>	<p>Retourne le nombre de milliseconde écoulées depuis la dernière mise sous tension de la carte.</p> <p>Exemple :</p> <pre>print Timer/1000 ; " secondes écoulées "</pre>
<b>Tick</b>	<p>Retourne l'état d'un compteur interne à la carte avec une résolution de 1/48000000s. Ce compteur peut être utilisé pour mesurer des temps avec précision.</p> <p>Attention, il s'agit d'un compteur sur 32 bits. Par conséquent, ce compteur repasse à 0 environ toutes les 89 secondes.</p> <p>Exemple :</p> <pre>Afficher la durée d'un pulse sur l'entrée IN1 do while IN(1)=0 loop TickDebut = Tick do while IN(1)=1 loop Ecart = Tick-TickDebut print "Temps = " ; Ecart/48000000 ; "s"</pre>
<b>Timerus</b>	<p>Retourner le temps écoulé en µs depuis le dernier appel à la commande ResetTimerus. Cette fonction permet la mesure précises de période de temps. La période mesurée entre l'appel à ResetTimerus et Timerus <b>ne doit pas excéder</b> les 89s</p> <p>Exemple :</p> <p>L'exemple suivant mesure le temps entre l'activation de la sortie 1 et le passage à 1 de l'entrée 1.</p> <pre>ResetTimerus OUT 1,1 do : loop until IN(1) Print "Temps d'activation = ", Timerus, "µs"</pre>
<b>GetTimer(VariableTimer)</b>	<p>Retourne le temps restant sur une temporisation initialisée avec la commande SetTimer. Lorsque le timer est écoulé, cette fonction retourne 0.</p> <p>Exemple :</p> <pre>SetTimer Tempo1, 100 """ if GetTimer(Tempo1) then out 1,1 endif</pre>
<b>Status1</b>	<p>Retourne les bits de status 0 à 15 de la carte (Voir plus loin le détails des bits de status)</p>
<b>Status2</b>	<p>Retourne les bits de status 16 à 31 de la carte (Voir plus loin le détails des bits de status)</p>

<b>Status3</b>	Retourne les bits de status 32 à 47 de la carte (Voir plus loin le détails des bits de status)
<b>Status4</b>	Retourne les bits de status 48 à 63 de la carte (Voir plus loin le détails des bits de status)
<b>StsBit( BitNo )</b>  BitNo : Numéro du bit de status de 0 à 63	Lecture de l'état d'un des bits de registre de la carte.  Exemple :  ' Arrêt axe X StopAxes 1 ' Attendre arrêt axe X do while StsBit( 22) =1 loop Activation sortie 1 lorsque X s'est arrête. Out 1,1  (Voir plus loin le détails des bits de status)
<b>Make32( Poids_fort, Poids_faible)</b>	Combinaison de 2 valeurs 16 bits en une valeur 32 bits signée  Exemple : value = Make32(getMBreg(5637), getMBreg(5638))
<b>MakeFloat(Poids_fort, Poids_faible)</b>	Combinaison de 2 valeurs 16 bits en une valeur de type float  Exemple : value = MakeFloat(getMBreg(5637), getMBreg(5638))



**Détails du registre de status (Status1 et Status2)**

B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16
<b>B Limit Switch</b>	<b>A Limit Switch</b>	<b>Z Limit Switch</b>	<b>Y Limit Switch</b>	<b>X Limit Switch</b>	<b>Axe B Async Moving</b>	<b>Axe A Async Moving</b>	<b>Axe Z Async Moving</b>	<b>Axe Y A sync Moving</b>	<b>Axe X Async Moving</b>	THCActivated	WaitInputError	WaitInputState	OverrideAllowed	Overridden	Reserve

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
EEWrite Error	EEWrite In Progress	<b>Probe Error</b>	<b>Probe In Progress</b>	<b>Homing Error</b>	<b>Homing In Progress</b>	Stroke Limit	<b>Board Locked</b>	<b>Emergency Stop</b>	Buffer Freezed	Buffer Empty	<b>Axe B Moving</b>	<b>Axe A Moving</b>	<b>Axe Z Moving</b>	<b>Axe Y Moving</b>	<b>Axe X Moving</b>

B0..B4 : Axes actuellement en cours de déplacement sur une interpolation linéaire.

B7 : Emergency Stop : État de l'entrée ENABLE/.

1 => L'entrée ENABLE n'est pas active. Carte verrouillée.

0 => L'entrée ENABLE est active (24V). La carte peut alors être utilisée.

B8 : Board Locked : Fonctionnement des sorties et des commandes d'axes verrouillées.

Tant que ce bit est actif, les mouvements d'axes et l'action sur les sorties sont verrouillés.

(Voir détail sur la commande N°66 : Reset et Ré-armement)

B10 : Homing In Progress : La commande Homing est en cours d'exécution.

B11 : Homing Error : La fonction Homing a échoué. Ce bit repasse automatiquement à 0 lors du lancement d'une nouvelle commande de Homing.

B12 : Probe In Progress : Fonction de palpé en cours.

B13 : Probe Error : La fonction de palpé a échoué. Ce bit repasse automatiquement à 0 lors du lancement d'une nouvelle commande de Palpé.

B22 à B26 : Un mouvement indépendant est en cours sur l'axe en question. Repasse à 0 lorsque le mouvement programmé est terminé ou interrompu par l'une des commandes de Stop.

B27 à B31 : Indicateur d'état des fins de courses

### Détails du registre de status étendu (Status 3 et Status4)

B62	B30	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48
<b>Reserve</b>	<b>Reserve</b>	<b>Reserve</b>	<b>Reserve</b>	Homming B Erreur	Homing B en cours	Homming A Erreur	Homing A en cours	Homming Z Erreur	Homing Z en cours	Homming Y Erreur	Homing Y en cours	Homming X Erreur	Homing X en cours	User prm changed	Modbus Received

B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
Reserve	Reserve	<b>Reserve</b>	<b>Reserve</b>	<b>Reserve</b>	<b>Reserve</b>	RecetteDataChanged	<b>Reserve</b>	<b>BasicRunning</b>	BasicRXOverrun	BasicTXOverrun	<b>Direction B</b>	<b>Direction A</b>	<b>Direction Z</b>	<b>Direction Y</b>	<b>Direction X</b>

B32..B36 : Direction actuelle des déplacements (0 si négatif, 1 si positif). Ces bits ne représentent pas la valeurs physique des sorties de direction (qui dépend des paramètres de sens de rotation) mais bien les sens de déplacement.

B37: Perte de caractères durant la transmission par l'interpréteur Basic

B38 : Perte de caractères durant la réception par le programme Basic

B39 : Programme Basic en cours d'exécution

B48 : Une commande Modbus a été reçue. Ce bit repasse automatiquement à 0 après lecture.

B49 : La mémoire EEPROM contenant les paramètres utilisateur a été modifiée. Ce bit repasse à 0 automatiquement à 0 après lecture.

B50 : Cycle de Homing en cours sur l'axe X (homing lancée par la commande basic Home)

B51 : Erreur détectée durant le cycle de homing sur l'axe X (homing lancée par la commande basic Home)

## 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 série de « If Etape=10 then... Else... Endif »

**Les instructions associées disponibles sont :**

**Select Case** Variable

**Case** Value

**Case** ValueFrom **to** ValueTo

**Case** Value **is** < Limite

**Case Else**

**End Select**

### Exemple d'utilisation

Supposons que nous souhaitons créer un cycle automate de 4 étapes (étape 0, 10, 20, et 30) dans lequel nous allons activer/désactiver la sortie 1 selon l'état de l'entrée 1, puis activer/désactiver la sortie 2 avec une temporisation de 1000ms.

Nous utiliserons par exemple une variable nommée « GCycle1 », qui prendra tour à tour la valeur d'étape en cours. Celle-ci sera réaffectée en fin de chaque étape, ce qui permettra de passer à la suivante au prochain tour de la boucle DO ... LOOP.

GCycle1 = 0      '*Initialisation du Cycle à l'étape 0*

DO

    Select Case GCycle1

    Case 0

        if IN(1) then

            out 1,1

            GCycle1 = 10

        endif

    Case 10

        if not IN(1) then

            out 1,0

            GCycle1 = 20

        endif

    Case 20

        out 2,1

        SetTimer tempo1, 1000

        GCycle1 = 30

    Case 30

        if GetTimer(tempo1)=0 then

            out 2,0

            GCycle1 = 0

        endif

    Case Else

        ? "Erreur de programmation"

    End Select

LOOP

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

## UTILISATION DES CARTES D'AXES INTERPCNC V2 ET V2.1D EN MODE DMX

### Introduction

Le DMX est un protocole de communication très répandu, en particulier dans le monde du spectacle, permettant les échanges d'informations entre différents appareils (console de commande, jeux de lumière, actionneurs motorisés, etc...)

Le DMX 512 est une norme dérivée du protocole Modbus, utilisant une liaison RS485, et permettant d'obtenir des valeurs de 0 à 255, sur 512 canaux numérotés de 1 à 512.

*Basées sur le protocole Modbus en RS485, les cartes InterpCNC 3 et 5 axes sont également capables de traiter des signaux DMX.*

*Ceci vous permet en élaborant un programme simple, de piloter/commander/régler différents types de matériels.*

### I) Pré-requis

- 1) Posséder une console DMX512.
- 2) Mettre à jour la carte InterpCNC à l'aide du firmware PLC spécifique DMX, disponible sur notre site dans cette archive:

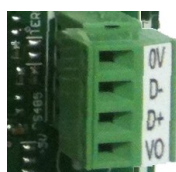
[https://www.soprolec.com/shop/fr/index.php?controller=attachment?id\\_attachment=79](https://www.soprolec.com/shop/fr/index.php?controller=attachment?id_attachment=79)

Dans le répertoire « Firmware » vous trouverez les instructions et les outils de mise à jour, et la dernière version du firmware PLC spécifique au mode DMX.

### II) Raccordement

Raccorder les signaux D+ (fil vert) et D- (fil jaune) issus de la prise DMX de votre console, sur les entrées D+ et D- de la carte InterpCNC :

Carte 5 axes V2.1D :

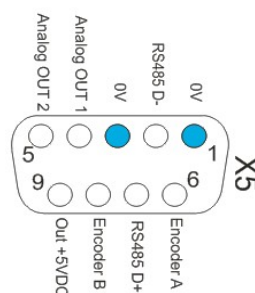


Carte 3 axes V2 : sur le connecteur X5

D+ **broche 7**, et D- **broche 2**



Micro Automate  
InterpCNC V2  
3 axes



### **III) Utilisation du DMX dans votre programme**

Les fonctions introduites par ce Firmware sont :

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

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

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

### **IV) Programme simple de démonstration**

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

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

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

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

NB : Le UserMem 0 est le premier des 16 registres utilisateurs en Ram. Nous l'utiliserons pour compter le nombre de trames reçues depuis le 1<sup>er</sup> démarrage du programme.

Pour faire un exemple simple de démonstration, la vitesse sera la valeur du canal DMX2 \* 10, soit de 0 à 2550 Hz

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

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

```

1 | Declaration des constantes
2 | *****
3 | Accel/decel des mouvement moteur
4 const ACCEL = 10 'KHz/s
5 ' Timeout reception DMX provoquant l'arret des mouvements et l'activation du frein
6 const TIMEOUT_DMX = 2000 ' (ms)
7
8 | Adresses DMX
9 const DMX_ON_OFF = 1 ' Adresses Marche/arrêt sortie OUT1
10
11 const DMX_V1 = 2 ' Adresses Vitesse moteur 1
12 const DMX_POS1 = 3 ' Adresses Position moteur 1
13
14 'Sauvegarde position et vitesse moteur 1
15 V1=0
16 POS1=0
17
18 | *****
19 ▼ Initialisation:
20 OUTALL 0
21 Pause 300
22
23 Unlock
24
25
26 ' Boucle programme principal
27 ▼ Do
28
29 ' L'entree ENABLE a ete coupee (ex : arrêt d'urgence)
30 if stsbit(8)=1 then goto Initialisation
31
32 ' Gestion timeout reception DMX
33 ▼ if IsDMXReceived then
34 SetUserMem 0, GetUserMem(0) + 1 ' Compteur reception DMX
35 DMXReceived = 1
36 TimerTimeoutDMX = Timer + TIMEOUT_DMX
37 TimeoutDMX = 0
38 else
39 DMXReceived = 0
40 endif
41
42 ' Detection d'un timeout de reception DMX
43 ▼ if Timer > TimerTimeoutDMX and not TimeoutDMX then
44 ? "Timeout DMX"
45 TimeoutDMX = 1
46 ' Action à faire si pas de reception DMX pendant plus de TIMEOUT_DMX(ms)
47 endif
48
49
50 ▼ if DMXReceived then ' Si nouvelles commandes DMX
51 ' Si canal DMX_ON_OFF > 127, activer sortie 1, sinon, désactiver sortie 1
52 if GetDMX(DMX_ON_OFF)>127 then
53 OUT 1, 1
54 else
55 OUT 1, 0
56 endif
57
58 ' Traitement des canaux 2 et 3 correspondant à Vitesse et position de l'axe 1
59 newV1 = GetDMX(DMX_V1)
60 newPOS1=GetDMX(DMX_POS1)
61 ▼ if NewV1 <> V1 or newPOS1 <> POS1 then
62 if V1 = 0 then
63 StopAxes 1
64 ▼ else
65 frequence = NewV1 * 10 ' transformation Valeur DMX 0..255 en fréquence de 0 à 2550Hz
66 Position = NewPOS1 * 100 ' transformation Valeur DMX 0..255 en pas moteur de 0 à 255000
67 moveaxe 1, ACCEL, frequence, ACCEL, Position
68 endif
69 V1 = NewV1
70 POS1 = NewPOS1
71 endif
72 endif
73
74 Loop

```

**Ligne 33 :**

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

**Ligne 43 :**

Traitement de l'éventualité d'un Time Out.

**Ligne 50 à 72:** Traitement de la trame DMX reçue :

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

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

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

Fin du programme.

## EXEMPLES DE PROGRAMMES

### ***Sauvegarde périodique des positions en EEPROM***

```

Dim PrevPosition(5) ' Tableau sauvegarde position précédente

'Restauration des positions sauvegardées
SetPos 1, GetEEData32(40) : PrevPosition(1) = GetEEData32(40)
SetPos 2, GetEEData32(41) : PrevPosition(2) = GetEEData32(41)
SetPos 3, GetEEData32(42) : PrevPosition(3) = GetEEData32(42)
SetPos 4, GetEEData32(43) : PrevPosition(4) = GetEEData32(43)
SetPos 5, GetEEData32(44) : PrevPosition(5) = GetEEData32(44)

' Mise en place sauvegarde periodique position
SetTick 1, 250, OnSavePositions

DO
  ' Application
LOOP

' Sauvegarde periodique des positions
OnSavePositions:
  for SaveAxe=1 to 5
    if (PrevPosition(SaveAxe)<>GetPos(SaveAxe)) then ' Si l'axe a bougé
      SetEEData32 40+SaveAxe-1, GetPos(SaveAxe)
      PrevPosition(SaveAxe) = GetPos(SaveAxe)
    endif
  next SaveAxe
iReturn

```

### ***Activer/désactiver un clignotant sur front montant d'une entrée***

```

DO
  ' Si front montant entrée 1, mettre en place un timer périodique de 500ms
  if (DFM(1)) then SetTick 1, 500, OnTimer500ms
  'Si Front descendant entrée 1, Arrêt du timer périodique 500ms
  if (DFD(1)) then
    SetTick 1, 0, 0      ' Arrêt du Timer
    OUT 5,0              ' Arrêt de la sortie
  endif
LOOP

OnTimer500ms :
  ' Changer l'état de la sortie 5
  OUT 5, not GETOUT(5)
iReturn

```



## ***Gestion de déplacement des axes par des entrées MARCHE et DIRECTION***

Dans cet exemple, nous utilisons des tableaux d'état pour gérer des cycles identiques mais indépendants sur les différents axes sans dupliquer le code pour chacun de ces axes.

*' Paramètres vitesse/accel/decel*

VITESSE1 = 10000 'Hz

VITESSE2 = 3500 'Hz

VITESSE3 = 17000 'Hz

ACCEL = 15 'Khz/s

DECEL = 15 'Khz/s

*'Affectation des entrées*

MARCHE1 = 1

DIRECTION1 = 2

MARCHE2 = 3

DIRECTION2 = 4

MARCHE3 = 5

DIRECTION3 = 6

*' Mise en tableaux des affectations*

dim MARCHE(3)

dim DIR(3)

dim VITESSE(3)

MARCHE(1) = MARCHE1

MARCHE(2) = MARCHE2

MARCHE(3) = MARCHE3

DIR(1) = DIRECTION1

DIR(2) = DIRECTION2

DIR(3) = DIRECTION3

VITESSE(1) = VITESSE1

VITESSE(2) = VITESSE2

VITESSE(3) = VITESSE3

*' Étapes Grafcet*

dim G(3)

do

*' ----- Combinatoire -----*

*' Déverrouillage automatique*

if (StsBit(8)=1) and StsBit(7)=0) then

Unlock

G(1) = 5

G(2) = 5

G(3) = 5

```

    ? "Déverrouillage automatique"
    Pause 100
endif

'Arrêt des séquences sur FD de l'Enable
if DFMBit(1,StsBit(7)) then
    ' Détection AU
    ? "Détection AU"
    G(1) = 0
    G(2) = 0
    G(3) = 0
    Pause 100
endif

' ----- Séquentiel -----

' Trois Grafjets identiques de gestion mouvement
for i=1 to 3
    if G(i)=0 then
        'Attente lancement
    elseif G(i)=5 then
        SetPos 1,0
        G(i)=10
    elseif G(i)=10 then
        if DFM(MARCHE(i)) then G(i)=50
    elseif G(i)=50 then
        if IN(DIR(i))=0 then
            Cible = 9999999
        else
            Cible = -9999999
        endif
        MoveAxe i, ACCEL, VITESSE(i), DECEL, Cible
        G(i) = 60
    elseif G(i)=60 then
        if IN(MARCHE(i))=0 then
            StopAxes (2 ^ (i-1))
            G(i) = 10
        endif
    endif
endif

next i
loop

```

## Déplacements des axes X et Y par un joystick

'AIN1 : Joystick axe X

'AIN2 : Joystick axe Y

'OUT1 : Carte prête

'Paramètres programme sauvegardés en EEPROM utilisateur

pResolutionX = GetEEDataFloat(0) 'pas/mm

pResolutionY = GetEEDataFloat(1) 'pas/mm

pCourseMaxiX = GetEEDataFloat(2) 'mm

pCourseMaxiY = GetEEDataFloat(3) 'mm

pVitesseMini = GetEEDataFloat(4) 'mm/s

pVitesseMaxi = GetEEDataFloat(5) 'mm/s

pAccel = GetEEDataFloat(6) 'mm/s<sup>2</sup>

pDecel = GetEEDataFloat(7) 'mm/s<sup>2</sup>

pJoyMinX = GetEEDataFloat(8)

pJoyCentreX = GetEEDataFloat(9)

pJoyMaxX = GetEEDataFloat(10)

pJoyMinY = GetEEDataFloat(11)

pJoyCentreY = GetEEDataFloat(12)

pJoyMaxY = GetEEDataFloat(13)

pJoyDeadZone = GetEEDataFloat(14)

' Conversion en unité d'axe

'ResolutionX = GetEEDataReal(0) 'pas/mm

'ResolutionY = GetEEDataReal(1) 'pas/mm

CourseMaxiX = pCourseMaxiX\*pResolutionX 'pas

CourseMaxiY = pCourseMaxiY\*pResolutionY 'pas

VitesseMini = pVitesseMini \* pResolutionX 'Hz

VitesseMaxi = pVitesseMaxi \* pResolutionY 'Hz

Accel = pAccel \* pResolutionX / 1000 'KHz/s<sup>2</sup>

Decel = pDecel \* pResolutionY / 1000 'KHz/s<sup>2</sup>

*'Calcul des plages de réglage par joystick*

*DVXMoinsParDAC = (VitesseMaxi - VitesseMini) / ( pJoyCentreX - pJoyMinX - pJoyDeadZone)*

*DVXPlusParDAC = (VitesseMaxi - VitesseMini) / ( pJoyMaxX - pJoyCentreX - pJoyDeadZone)*

*DVYMoinsParDAC = (VitesseMaxi - VitesseMini) / ( pJoyCentreY - pJoyMinY - pJoyDeadZone)*

*DVYPlusParDAC = (VitesseMaxi - VitesseMini) / ( pJoyMaxY - pJoyCentreY - pJoyDeadZone)*

*Initialisation:*

*OUT 1,0*

*'déverrouillage de la carte*

*unlock*

*if stsbit(8)=1 then ' test bit carte verrouillée*

*pause 50*

*goto Initialisation*

*endif*

*OUT 1,1 ' Indique que la carte est déverrouillée*

*DoUpdateSpeed = 1 ' Pour initialisation des vitesse*

*' Activation traitement périodique de gestion vitesse*

*settick 1,50,ontick1*

*do*

*' L'entrée ENABLE à été coupée*

*if stsbit(8)=1 then goto Initialisation*

*' Pour affichage position*

*SetUserMemF 0,GetPos(1)/pResolutionX*

*SetUserMemF 1,GetPos(2)/pResolutionY*

*if (DoUpdateSpeed=1) then ' A intervalle régulier (suivant ontick1)*

*GoSub UpdateVitesse*

*if (VitesseX>0) then*

*MoveAxe 1,Accel, VitesseX, Decel, CourseMaxiX*

*elseif (VitesseX<0) then*

*MoveAxe 1,Accel, -VitesseX, Decel, 0*

*else*

*StopAxes 1*

*endif*

*if (VitesseY>0) then*

*MoveAxe 2,Accel, VitesseY, Decel, CourseMaxiY*

*elseif (VitesseY<0) then*

*MoveAxe 2,Accel,-VitesseY, Decel, 0*

*else*

*StopAxes 2*

*endif*

*DoUpdateSpeed = 0*

*endif*

*loop*

*UpdateVitesse:*

*'Calcul vitesse de déplacement X*

*NewAIN1 = AIN(1)*

*if (NewAIN1<10) then ' Cas ou le joystick est débranché*

*VitesseX = 0*

*elseif (abs(NewAIN1-pJoyCentreX) <pJoyDeadZone) then*

*VitesseX = 0*

*elseif (NewAIN1>pJoyCentreX) then*

*VitesseX = (NewAIN1 - pJoyCentreX - pJoyDeadZone) \* DVXPlusParDAC*

*else*

*VitesseX = -(pJoyCentreX - pJoyDeadZone - NewAIN1) \* DVXMoinsParDAC*

*endif*

*'Calcul vitesse de déplacement Y*

*NewAIN2 = AIN(2)*

*if (NewAIN2<10) then ' Cas ou le joystick est débranché*

*VitesseY = 0*

*elseif (abs(NewAIN2-pJoyCentreY) <pJoyDeadZone) then*

*VitesseY = 0*

*elseif (NewAIN2>pJoyCentreY) then*

*VitesseY = (NewAIN2 - pJoyCentreY - pJoyDeadZone) \* DVYPlusParDAC*

*else*

*VitesseY = -(pJoyCentreY - pJoyDeadZone - NewAIN2) \* DVYMoinsParDAC*

*endif*

*SetUserMemF 2,VitesseX/pResolutionX*

*SetUserMemF 3,VitesseY/pResolutionY*

*Return*

*' Timer periodique de gestion de vitesse*

*ontick1:*

*DoUpdateSpeed = 1*

*ireturn*

**Déplacements des axes X et Y par l'entrée codeur**

' Exemple gestion manivelle électronique sur entrée codeur

' Entrées 9, 10 et 11 pour sélectionner l'axe à déplacer

' Entrée codeur pour déplacer un axe

' Initialisation

Etape = 0

SetPrm 126, 2 ' Entrée codeur mode 4X

do

if Etape = 0 then ' Initialisation

Ratio = 4 ' Manivelle 400 pulse/tour, moteur 1600 pas -> un tour manivelle pour 1 tour moteur

Etape = 10

elseif Etape = 10 then

if DFM(9) then ' Si front sur entrée 9, mouvements X

Etape = 20

Entree = 9

Axe = 1

elseif DFM(10) then ' Si front sur entrée 10, mouvements Y

Etape = 20

Entree = 10

Axe = 2

elseif DFM(11) then ' Si front sur entrée 11, mouvements Z

Etape = 20

Entree = 11

Axe = 3

endif

elseif Etape=20 then

SetCnt 0,GetPos(Axe) / Ratio ' Initialisation entrée codeur

Pos1 = GetEncoder

Etape = 30

elseif Etape = 30 then

Pos2 = getencoder

dmove = Pos2 - Pos1

Pos1 = Pos2

if dmove<>0 then ' Déplacement si manivelle bougée

moveaxe Axe, 100, 25000,100, getencoder\*ratio 'getpos(1)+dmove\*10

endif

if In(Entree)=0 then Etape = 10 ' Arrêt si entrée validation repasse à 0

endif

loop

**Asservissement analogique d'un axe avec PID et entrée codeur**

'On utilise le générateur de profile de l'axe X  
 'et l'asservissement se fait entre la position du compteur X et  
 'la valeur du retour codeur

'Initialisation position axe X  
 SetPos 1,GetEncoder  
 unlock

'Chargement de paramètres PID2F  
 gosub UpdateParameters

' Lancement boucle de régulation  
 SetTick 1, h\*1000, OnTick1

do  
 'Mise à jour des paramètres si modification  
 SetUserMem 0,GetEncoder – GetPos(1)  
 if isMBPrmChanged then'StsBit(49)=1 then  
 Gosub UpdateParameters  
 endif  
 Loop

UpdateParameters:

? "Update parameters"  
 Kp= GetEEDataFloat(0) ' Gain statique Kp  
 Ti= GetEEDataFloat(1) ' Temps intégrateur (s) Si grande valeur, peu d'effet  
 Td = GetEEDataFloat(2) 'Temps dérivateur (s) Si nul, pas d'effet  
 N = GetEEDataFloat(3) 'Filtre de dérivée (0.5 à 10)  
 h= GetEEDataFloat(4) 'Temps d'échantillonnage en seconde  
 '? "Kp=",kp, " ti=",ti," td=",td," N=",n," h=",h  
 'SetPID2FParameters Kp,Ti,Td,N,h,Mini, Maxi, Offset  
 SetPID2FParameters Kp,Ti,Td,N,h,0, 1023,0  
 SetTick 1, h\*1000, OnTick1  
 return

'Appel périodique correcteur PID2F

OnTick1:

'Calcul Erreur  
 e0 = GetPos(1) – GetEncoder  
 ' Calcul consigne  
 u0= PID2F(e0)  
 'Envoie commande sur sortie analogique  
 SetAna 1, u0  
 ireturn



## HISTORIQUE DES MODIFICATIONS

- 30/03/2021 : Ajout informations sur l'utilisation de l'instruction **Select Case** (depuis le firmware 5.32). Exemple de création de **Grafcet**
- 04/12/2020 : Ajout informations sur la commande **OUTTTL**, et l'utilisation des sorties TTL (depuis le firmware 5.29)
- 15/09/2020 : Ajout informations sur l'utilisation du Mode **DMX**.
- 03/03/2020 : Ajout commande **SetCaptureInt** (interruption sur position capturée)  
Ajout des types 4, 5, et 6, pour le déclenchement des interruptions de **SetInputInt**  
- A partir du firmware 5.34
- 04/05/2016 : Ajout commande Home (cycle d'initialisation d'axes)
- 01/09/2014 : Ajout exemple programme Basic pour régulateur PID
- 04/07/2014 : Ajout exemple utilisation entrée codeur avec manivelle électronique
- 09/04/2014 : Correction information fonction **IsMBPrmChanged**
- 08/12/2013 : Ajout d'exemple de programme en Basic
- 28/11/2013 : Ajout information commande **Probe**  
Ajout commande **GetSys(xxx)** - A partir du firmware 5.18
- 02/07/2013 : Ajout fonction **GetDIPSwitch**.