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, actuateurs 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 Modus 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

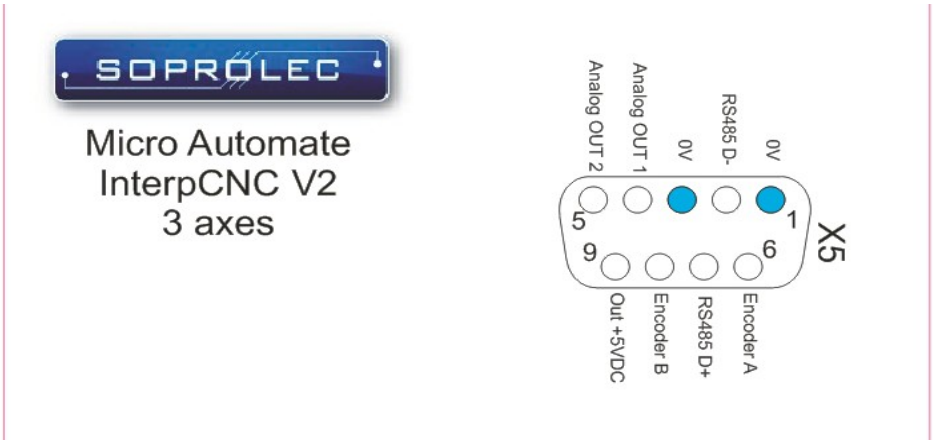
Dans le répertoire « Firmware » vous trouverez les instructions et les outils de mise à jour, et la dernière version du firmware PLC spécifique au mode DMX.

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**



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 souhaitions 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 1er 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.

```
' Declaration des constantes
' *************************
' Accel/decel des mouvement moteur
const ACCEL = 10    'KHz/s
' Timeout reception DMX provoquant l'arret des mouvements et l'activation du frein
const TIMEOUT_DMX = 2000     ' (ms)

' Adresses DMX
const DMX_ON_OFF = 1       ' Adresses Marche/arrêt sortie OUT1

const DMX_V1   = 2         ' Adresses Vitesse moteur 1
const DMX_POS1 = 3         ' Adresses Position moteur 1

'Sauvegarde position et vitesse moteur 1
V1=0
POS1=0

' ************************************************************
Initialisation:
   OUTALL 0
   Pause 300

   Unlock


' Boucle programme principal
Do

   ' L'entree ENABLE a ete coupee (ex : arrêt d'urgence)
   if stsbit(8)=1 then goto Initialisation

   ' Gestion timeout reception DMX
   if IsDMXReceived then
     SetUserMem 0, GetUserMem(0) + 1    ' Compteur reception DMX
     DMXReceived = 1
     TimerTimoutDMX = Timer + TIMEOUT_DMX
     TimeoutDMX = 0
   else
     DMXReceived = 0
   endif

   ' Detection d'un timeout de reception DMX
   if Timer > TimerTimoutDMX and not TimeoutDMX then
     ? "Timeout DMX"
     TimeoutDMX = 1
     ' Action à faire si pas de reception DMX pendant plus de TIMEOUT_DMX(ms)
   endif


   if DMXReceived then     ' Si nouvelles commandes DMX
     ' Si canal DMX_ON_OFF > 127, activer sortie 1, sinon, désactiver sortie 1
     if GetDMX(DMX_ON_OFF)>127 then
       OUT 1, 1
     else
       OUT 1, 0
     endif

     ' Traitement des canaux 2 et 3 correspondant à Vitesse et position de l'axe 1
     newV1 = GetDMX(DMX_V1)
     newPOS1=GetDMX(DMX_POS1)
     if NewV1 <> V1 or newPOS1 <> POS1 then
       if V1 = 0 then
         StopAxes 1
       else
         frequence = NewV1 * 10     ' transformation Valeur DMX 0..255 en fréquence de 0 à 2550Hz
         Position = NewPOS1 * 100    ' transformation Valeur DMX 0..255 en pas moteur de 0 à 255000
         moveaxe 1, ACCEL, frequence, ACCEL, Position
       endif
       V1 = NewV1
       POS1 = NewPOS1
     endif
   endif

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.