SOPROLEC
www.soprolec.com
ZAC DE L'EPINE
72460 SAVIGNE L'EVEQUE
Tél. : +33 (0)2 43 76 44 76

Librairie ICNC Labview

Sommaire

Introduction

Suite à plusieurs demandes concernant la compatibilité des cartes InterpCnC avec le logiciel de programmation Labview, une bibliothèque USB a été conçue pour permettre la création et l'utilisation d'application Labview pour les cartes InterpCnC.

Cette bibliothèque regroupe une grande partie des fonctionnalités de la carte InterpCnC et permet un pilotage complet de la carte et une grande marge de manœuvre dans l'accès aux paramètres de celle-ci ainsi qu'à ses fonctionnalités.

Vous pourrez retrouver dans ce document la marche à suivre pour installer la librairie USB de la carte ICNC.
Les fonctions de cette librairie (ses VIs) sont décrites, et, pour exemple, une application complète est réalisée avec cette librairie. Son utilisation est donc expliquée en détail.

Les différents VI , projets et tests ont été effectués avec Labview 2012.

Installation de la Librairie ICNC

Pour utiliser la librairie ICNC il vous faut d'abord l'installer dans les dossier de votre Labview :

Vous devez suivre un chemin semblable à :
C:/ Program Files(x86)/National Instruments/Labview 2012/user.lib/

pour mettre le fichier ICNC contenant la librairie.

Vous pouvez aussi installer la librairie comme Nouvelle palette Labview en la copiant dans le fichier :
C:/ Program Files(x86)/National Instruments/Labview 2012/menus/Categories/

et elle apparaîtra ainsi comme une palette à part entière facilitant donc son accès.

Vous devriez maintenant pouvoir, en relançant complètement labview, avoir accès à la librairie ICNC dans tous vos projets Labview dans la palette bibliothèque utilisateur ou directement en palette si vous l'avez installée comme telle.

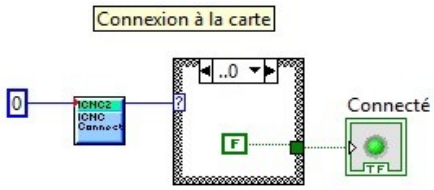
Fonctions de la Librairie ICNC

Nous allons dans ce chapitre donner des précisions sur les différentes fonctions de la librairie.
Certaines de ces fonctions telles que la fonction Connect() et la fonction ErrorReset() seront obligatoires dans chacune de vos applications.

Connect / Disconnect

Fonction Connect()

Cette fonction permet de connecter l'application à la carte, cette fonction sera nécessaire dans toutes vos applications.
Vous pouvez tester le retour de cette fonction pour, comme ici dans l'exemple, afficher si la connexion a réussi via une LED. Ici les Paramètres ne sont pas nécessaires.

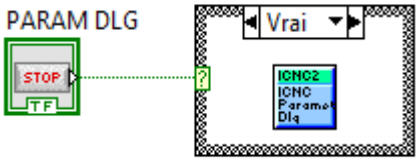


Fonction Disconnect()

Cette fonction permet de déconnecter la carte actuellement connectée. Ce n'est pas obligatoire si votre application est la seule à utiliser votre carte. Cependant, si vous voulez utiliser plusieurs applications, vous devrez vous déconnecter afin de pouvoir vous reconnecter dans une autre application. Pas de paramètres pour cette fonction.

Dlgs

Plusieurs fonctions DLG sont à votre disposition.
Ces fonctions vous permettent d'afficher des fenêtres affichant des paramètres, des informations et les statuts de la carte actuellement connectée.
Ces fonctions sont simples à utiliser dans labview car il vous suffit d'appeler la fonction via un bouton ou une action, sans paramètre pour les fonctions.



Fonction ParamDLG()

Cette fonction permet d'afficher une fenêtre existant dans TestCenter regroupant de nombreux paramètres et informations sur la carte qui est actuellement connectée. Pas de paramètres pour cette fonction.

Fonction BasicControlDLG()

Cette fonction ouvre l'interpréteur Langage BASIC de la carte. Cette interpréteur vous permet d'embarquer un programme sur la carte elle même, de sauvegarder ce programme sur la carte ou sur votre machine. Pas de paramètres pour cette fonction.

Fonction PlasmaSettingDLG()

Cette fonction ouvre une fenêtre de paramètres pour l'utilisation de torche plasma piloté par la carte InterpCnC. Ici les Paramètres ne sont pas nécessaires.

Fonction UserEEPROMDLG()

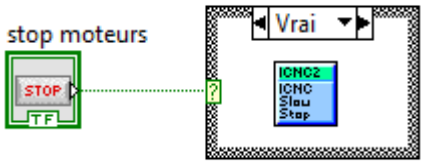
Cette fonction ouvre un gestionnaire de la mémoire EEPROM affichable en 8bits, 16bits, 32bits ou en réel. Un seul paramètre pour cette fonction qui correspond au taux de rafraîchissement de l'affichage de la mémoire.

Fonction UserMBBitDLG()

Cette fonction ouvre un gestionnaire de mémoire par bit disponible pour l'utilisateur.
Pas de paramètres pour cette fonction.

Stop

Les fonctions Stop sont les fonctions qui vont vous permettre d'arrêter les mouvements sur les axes de la carte. Ces fonctions sont simples d'utilisation car elles ne requièrent aucun paramètre, elles ont juste à être appelées par un bouton par exemple.



Fonction StopMotorsAllAndClear()

Cette fonction permet un arrêt rapide des moteurs et le verrouillage de la carte. Elle est particulièrement intéressante pour un arrêt d'urgence dans votre application.
Pas de paramètres pour cette fonction.

Fonction SlowStopAllAndClear()

Cette fonction permet d'arrêter les moteurs lentement et de verrouiller la carte, cela permet d'avoir une décélération avant l'arrêt des moteurs. Pas de paramètres pour cette fonction.

Fonction SlowStopMotors()

Cette fonction permet d'arrêter les moteurs après une décélération sans verrouiller la carte.
Un seul paramètre pour cette carte qui est l'axe à arrêter.

Read

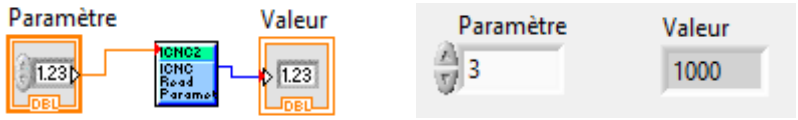
Les fonctions Read vont vous permettre de lire le contenu de la mémoire à disposition dans la carte.

Fonction ReadEEPROM

Cette fonction permet de lire le contenu de la mémoire EEPROM présente sur la carte InterpCnC. Trois paramètres pour cette fonction ; l'adresse de départ de la lecture dans la mémoire, la taille de la lecture et un pointeur sur lequel sera envoyer la valeur lue.

Fonction ReadParameter()

Cette fonction vous permet de lire un paramètre spécifique de la carte.



Les Paramètres de cette fonction sont l'ID du paramètre dans la mémoire et un pointeur sur lequel sera envoyée la valeur lue.

Fonction ReadUserMem()

Cette fonction permet de lire le contenu de la mémoire Utilisateur mise à votre disposition sur la carte InterpCnC. Deux paramètres pour cette fonction : le numéro de la mémoire et un pointeur sur lequel sera envoyer la valeur lue.

Write

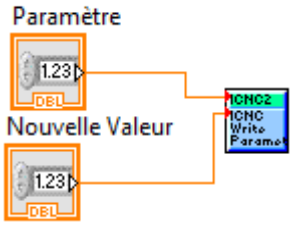
Les fonctions Write vont vous permettre d'écrire dans les mémoires disponibles ou dans les paramètres de la carte ou directement les positions de chaque axe.

Fonction WriteEEPROM()

Cette fonction vous permet d'écrire dans la mémoire EEPROM existante sur la carte.
Il y a quatre paramètres pour cette fonction: l'adresse de départ de l'écriture, la taille de l'écriture, la valeur à écrire et le temps à attendre pour compléter l'écriture.

Fonction WriteParameter()

Cette fonction vous permet d'écrire un paramètre spécifique de la carte.
Il y a quatre paramètres pour cette fonction: l'adresse de départ de l'écriture, la valeur à écrire, le temps à attendre pour compléter l'écriture, et un timeout pour que la fonction ne soit pas bloquante en cas de problème.



Fonction WriteUserMemBuf()

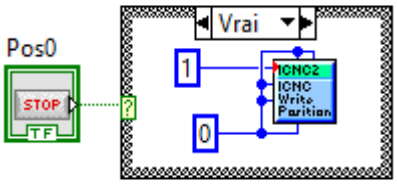
Cette fonction permet d'écrire dans le buffer d'écrite de la mémoire utilisateur.
Il y a deux paramètres pour cette fonction : le numéro de la mémoire et la valeur à écrire dans celle-ci.

Fonction WriteUserMem()

Cette fonction permet d'écrire dans la mémoire Utilisateur.
Il y a deux paramètres pour cette fonction : le numéro de la mémoire et la valeur à écrire dans celle-ci.

Fonction WritePositions()

Cette fonction permet d'écrire directement la position actuelle de chaque axe de la carte. Cette fonction peut notamment être utile pour définir à un moment donné que la position actuelle devient la position 0. Les paramètres de la fonction sont: les axes dont la position doit être écrite, et la valeur à associé à chacun d'entre eux.



Set

Les fonctions Set vont vous permettre de définir l'état de sorties ou de paramètres de la carte.

Fonction SetAnalogBuf()

Cette fonction permet de changer l'état d'une sortie analogique, cette commande est bufferisée. Il y a deux paramètres pour cette fonction : Le numéro de la sortie et la valeur de celle-ci.

Fonction SetAnalog()

Cette fonction permet de changer l'état d'une sortie analogique. Il y a deux paramètres pour cette fonction : Le numéro de la sortie et la valeur de celle-ci.

Fonction SetOutputAllBuf()

Cette fonction permet de définir l'état de toutes les sorties de la carte en même temps, cette commande est bufferisée. Il y a un paramètre pour cette fonction : la valeur à assigner à chaque sortie.

Fonction SetOutputAll()

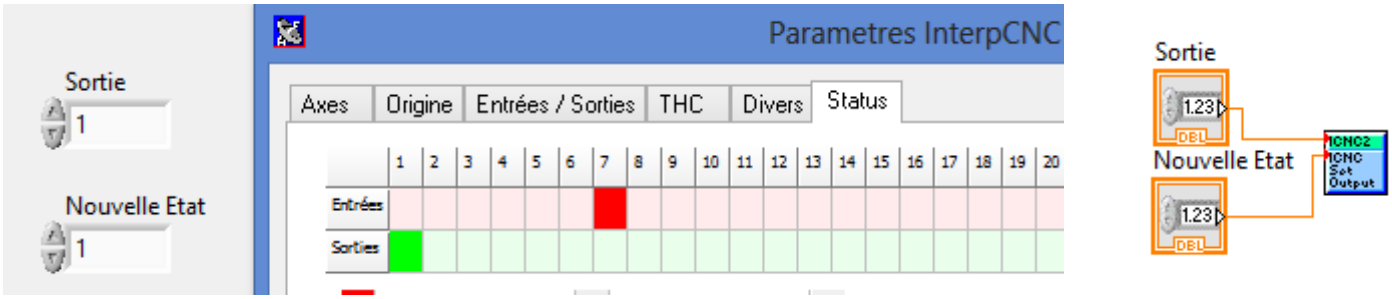
Cette fonction permet de définir l'état de toutes les sorties de la carte en même temps. Il y a un paramètre pour cette fonction : la valeur à assigner à chaque sortie.

Fonction SetOutputBuf()

Cette fonction permet de définir l'état d'une sortie de la carte, cette commande est bufferisée. Il y a deux paramètres pour cette fonction : Le numéro de la sortie et la valeur de celle-ci.

Fonction SetOutput()

Cette fonction permet de définir l'état d'une sortie de la carte. Dans l'exemple avec l'aide de la fonction ParameterDLG, on a bien une sortie 1 à l'état 1. Il y a deux paramètres pour cette fonction : Le numéro de la sortie et la valeur de celle-ci.



Fonction SetOverrideStateBuf()

Cette fonction permet d'activer ou désactiver le mode de sur-vitesse, cette commande est bufferisée. Il y a un seul paramètre pour cette fonction : l'état de sur-vitesse.

Fonction SetOverrideValue()

Cette fonction permet de définir la valeur de sur-vitesse associé à la carte. Il y a un seul paramètre pour cette fonction : la valeur de sur-vitesse.

Get

Les fonctions Get vont vous retourner des valeurs de paramètres de la carte.

Fonction GetAnalog()

Cette fonction retourne l'état de la sortie analogique sélectionnée.
Il y a deux paramètres pour cette fonction : le numéro de sortie analogique, et un pointeur sur lequel sera retourné la valeur de cette-ci.

Fonction GetBoardStatus()

Cette fonction renvoie le registre de statuts de la carte. Il y a deux paramètres pour cette fonction : le type de statuts et un pointeur sur lequel sera retourné le statuts.

Fonction GetBufferStatus()

Cette fonction retourne l'état du buffer de la carte. Il y a un paramètre pour cette fonction, c'est l'espace libre dans le buffer.

Fonction GetExecTime()

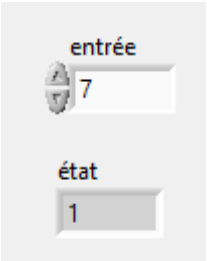
Cette fonction retourne le temps passé en déplacement. Il y a un paramètre sur cette fonction, c'est un pointeur sur lequel sera retourné le temps passé en déplacement.

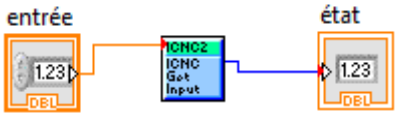
Fonction GetInputAll()

Cette fonction retourne l'état de toutes les entrées de la carte. Il y a un paramètre sur cette fonction, c'est un pointeur sur lequel sera retourné l'état des entrées.

Fonction GetInput()

Cette fonction retourne l'état de la sortie sélectionnée en paramètre de la fonction.
Comme on a pu le voir dans SetOutput l'entrée 7 était à 1, avec GetInput on a donc la valeur 1 qui est retournée par la fonction. Il y a deux paramètres sur cette fonction : le numéro de l'entrée et un pointeur sur lequel sera retourné l'état des entrées.





Fonction GetOutputAll()

Cette fonction retourne l'état de toutes les sorties de la carte. Il y a un paramètre sur cette fonction, c'est un pointeur sur lequel sera retourné l'état des sorties.

Fonction GetOverrideValue()

Cette fonction retourne la valeur de sur-vitesse associée à la carte. Il y a un paramètre sur cette fonction, c'est un pointeur sur lequel sera retourné la valeur de la sur-vitesse.

Fonction GetSysInfo()

Cette fonction retourne les informations système relatives à la carte.
Il y a deux paramètres pour cette fonction : le type d'information système et un pointeur qui recevra l'information.

Move

Les fonctions Move vont vous permettre d'effectuer des déplacements sur les différents axes de la carte. Vous pouvez trouver un exemple d'utilisation de ces fonctions dans le projet exemple, à la suite de la description des fonctions de la librairie dans ce document.

Fonction MoveArcXYBuf()

Cette commande permet de lancer un déplacement sur les Axes X et Y simultanément. Cette commande est bufferisée. Il y a plusieurs paramètres : la fréquence d'accélération / décélération, la cible de l'axe Y, le sens, le centre I, le centre J, la cible de l'axe X, la fréquence de start/stop et la vitesse.

Fonction MoveProfileAbsAsync()

Cette fonction permet de lancer un déplacement absolu en profil de vitesse asynchrone. Il y a plusieurs paramètres : l'axe, la fréquence de start/stop, l'accélération, la décélération, la vitesse et la position.

Fonction MoveProfileAbsbuf()

Cette fonction permet de lancer un déplacement absolu en profil de vitesse, cette commande est bufferisée. Il y a plusieurs paramètres : les axes, la vitesse, la position X, la position Y, la position Z, la position B, la position A et l'espace buffer requis.

Fonction MoveProfileRelAsync()

Cette fonction permet de lancer un déplacement relatif en profil de vitesse asynchrone. Il y a plusieurs paramètres : l'axe, la fréquence de start/stop, l'accélération, la décélération, la vitesse et la distance.

Fonction MoveProfileRelBuf()

Cette fonction permet de lancer un déplacement relatif en profil de vitesse. Cette commande est bufferisée. Il y a plusieurs paramètres : la vitesse, le déplacement X, le déplacement Y, le déplacement Z, le déplacement B, le déplacement A et l'espace buffer requis.

Fonction MoveSpeedAbsBuf()

Cette fonction permet de lancer un déplacement en vitesse constante ,ce mouvement est absolu et asynchrone, cette commande est bufferisée. Il y a plusieurs paramètres : les axes, la vitesse, le déplacement X, le déplacement Y, le déplacement Z, le déplacement B, le déplacement A et l'espace buffer requis.

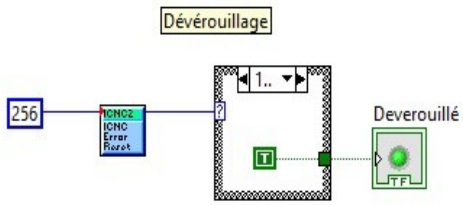
Fonction MoveSpeedRelBuf()

Cette fonction permet de lancer un déplacement en vitesse constante ,ce mouvement est relatif et asynchrone, cette commande est bufferisée. Il y a plusieurs paramètres : la vitesse, le déplacement X, le déplacement Y, le déplacement Z, le déplacement B, le déplacement A et l'espace buffer requis.

Autres Fonctions

Les autres fonctions regroupent les fonctions n'appartenant à aucune des catégories déjà définies . Vous serez cependant amené à utiliser certaines d' entre elles, tels que ErrorReset , indispensable à chaque application et MachineHome qui vous sera très utile pour les prises d'origine.

Fonction ErrorReset()



Cette fonction permet de remette à 0 les compteurs d'erreurs et aussi de remettre à 0 le bit de blocage de la carte. Cette fonction sera indispensable dans la plupart de vos applications. Vous pouvez, comme dans l'exemple, tester la réussite de la fonction et l'afficher via une LED. Le Paramètres 256 permet de sélectionner le bit de verrouillage de la carte. Le paramètre de la fonction est le bit d'erreur à reset.

Fonction MachineHome()

Cette fonction permet de réaliser la prise d'origine des différents axes de la carte. Les paramètres de la fonction sont : les axes et leurs valeurs maximales de déplacement.

Fonction Probe()

Cette fonction permet de réaliser un déplacement sur l'axe sélectionné jusqu'à un capteur. Il y a plusieurs paramètres à cette fonction : l'axe, la direction, le numéro de l'entrée associé au capteur, la valeur de cette entrée, la vitesse,l'accélération, la décélération et la course maximale à effectuer.

Fonction UpdateSpeed()

Cette fonction permet de mettre à jour la fréquence (et donc la vitesse) associé à l'axe sélectionné. Il y a deux paramètres : l'axe et la nouvelle fréquence à lui affecter.

Fonction LockBoard()

Cette fonction permet de verrouiller la carte actuellement connectée. Il n'y a pas de paramètres pour cette fonction.

Fonction ExtSPIControl()

Cette fonction permet le contrôle de bus SPI via un connecteur optionnel sur la carte. Il y a trois paramètres : le compte de byte envoyé, un pointeur sur le buffer envoyé et un pointeur sur le buffer reçu.

Fonction FlushLocalBuf()

Cette fonction permet de vider le buffer local. Elle est notamment utile après les commandes Move bufferisées. Pas de paramètres pour cette fonction.

Fonction FreezeBuf()

Cette fonction gèle le traitement du buffer jusqu'à sont remplissage ou l'appel de la fonction Unfreeze(). Pas de paramètres pour cette fonction.

Fonction ProbeAndReadBack()

Cette fonction lance un déplacement jusqu'à atteindre un capteur et retourne la position de celui-ci. Il y a plusieurs paramètres à cette fonction : l'axe, la direction, le numéro de l'entrée associé au capteur, la valeur de cette entrée, la vitesse, l'accélération, la décélération, la course maximale à effectuer et la position de probe.

Fonction Unfreeze()

Cette fonction dégèle le traitement du buffer précédemment gelé par la fonction Freezebuf(). Pas de paramètre pour cette fonction.

Fonction WaitBuf()

Cette fonction permet de définir une temporisation bufferisée. Un seul paramètre pour cette fonction : le temps à attendre.

Fonction WaitInputStateBuf()

Cette fonction permet une attente d'état d'entrée dans le buffer. Il y a plusieurs paramètres pour cette fonction : le numéro de l'entrée, la valeur de l'entrée, le timeout, le locklferroret le clearbufferlferror.

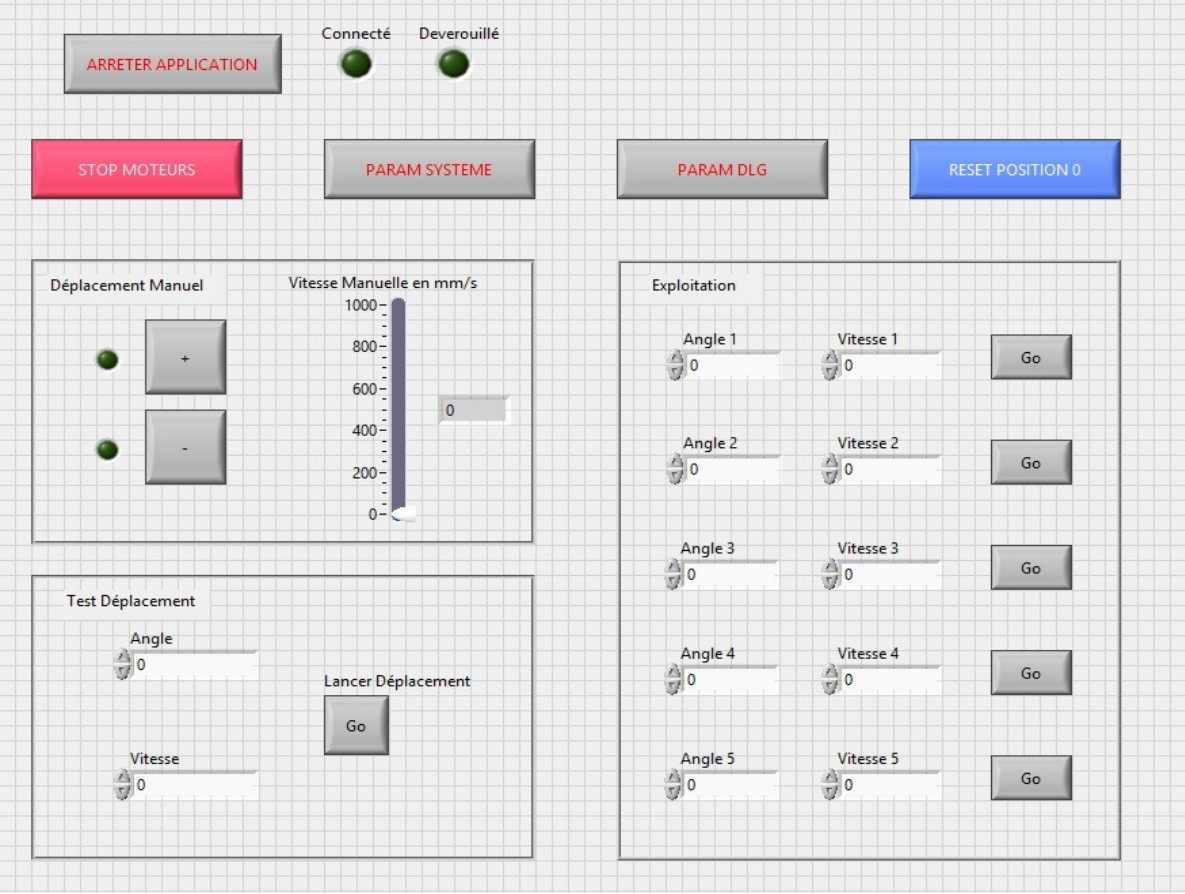
Pour certaines fonctions vous devrez indiquer un axe en paramètre, voici comment les indiquer correctement aux fonctions :
Axe X → 1, Axe Y → 2, Axe Z → 4, Axe A → 8, Axe B → 16 .

Projet Exemple Utilisation Librairie ICNC

Un exemple d'utilisation de la librairie existe. Cet exemple est basé sur un projet de tournette universelle pilotée par une carte InterpCnC. Cette tournette devait être pilotable en angle et vitesse. Grâce à ces valeurs un déplacement était calculé. Les déplacements devaient pouvoir être testés, soit manuellement soit en angle et vitesse et l'application devait disposer d'une partie exploitation permettant de rentrer plusieurs déplacements pour les exécuter à la suite.
Pour ce faire, une application Labview a été réalisée pour répondre à ce cahier des charges.

Voici l'interface principale réalisée :

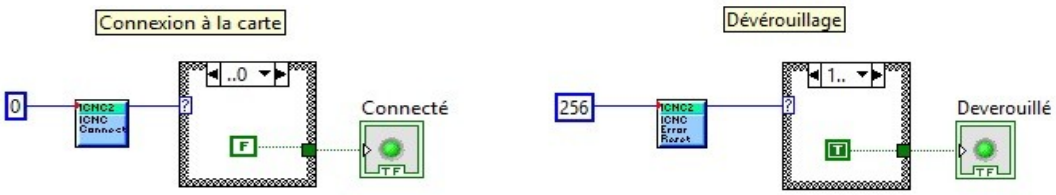
VI Principal



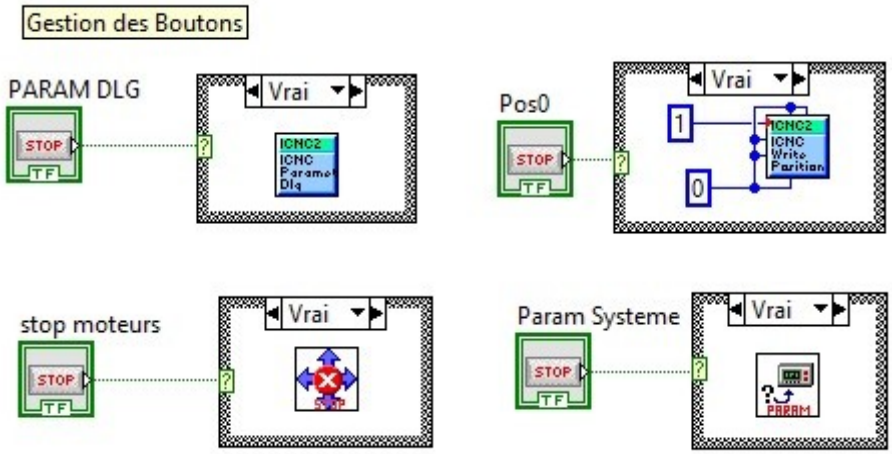
Plusieurs Vis supplémentaires ont été créés :

- un VI pour la configuration des paramètres de la tournette.
- un VI pour le bouton Stop Moteurs pour avoir un stop personnalisé.
- un VI Move pour effectuer les conversions nécessaires entre les paramètres que rentre l'utilisateur et le type de valeurs à rentrer dans la fonction move utilisée.
- une variable globale pour accéder aux paramètres de la tournette dans les autres VI principalement dans le VI move pour effectuer les conversions.

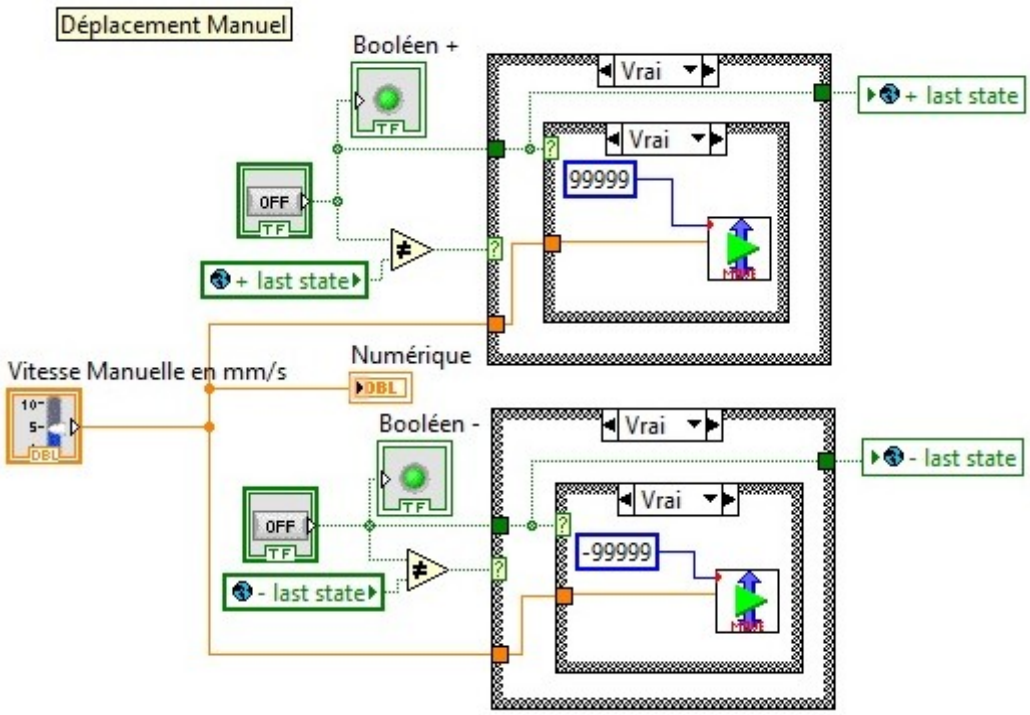
Le diagramme de cette interface principale se décompose de la façon suivante :



Tout d'abord, afin de pouvoir interagir avec la carte connectée en USB il faut lancer la fonction connect() dont vous trouverez les descriptions dans les fonctions de la librairie ICNC.
Il faut ensuite utiliser la fonction ErrorReset() pour déverrouiller la carte afin que la carte puisse exécuter ce qui lui ai demandé.



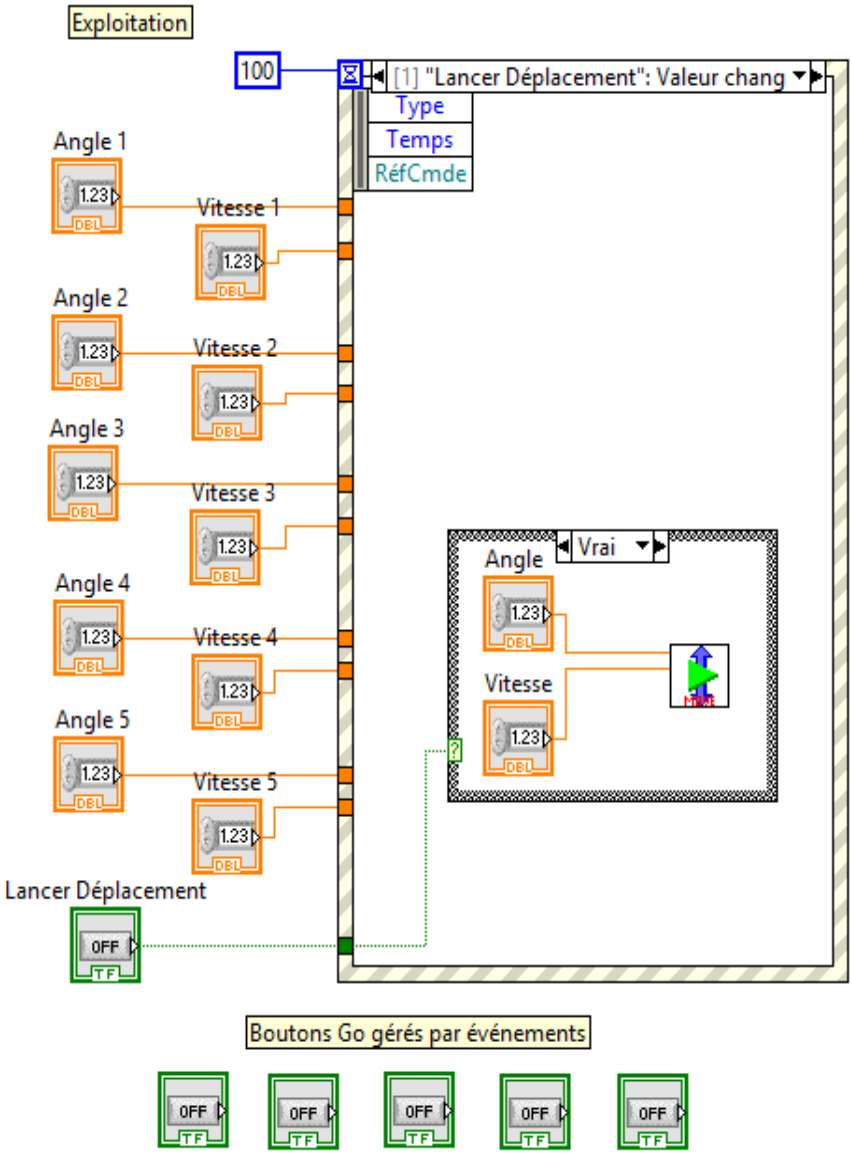
Ensuite, voici comment les boutons, param dlg, reset position 0, stop moteurs et Param systeme sont gérés. Chaque bouton est testé et pour chacun une fonction de la librairie ou un VI en utilisant une est lancé.



Voici comment sont gérées les commandes manuelles. Celle-ci sont assez complexes car, pour des raisons de sécurité, les boutons + et – produisent un déplacement sur l'axe de la tournette et lorsque le bouton est relâché, le déplacement doit absolument s'arrêter. Pour ce faire, nous avons donc du traiter l'ancien état du bouton pour avoir un déplacement et stop lancé de façon unique à chaque appui sur les boutons. Les valeurs 99999 et -99999 permettent d'avoir un déplacement quasi sans fin , jusqu'à ce qu'un stop arrête celui-ci.

Vient ensuite la gestion de la partie exploitation.
Une gestion événementielle est utilisée car ces boutons ne sont pas faits pour être lancés en simultané.

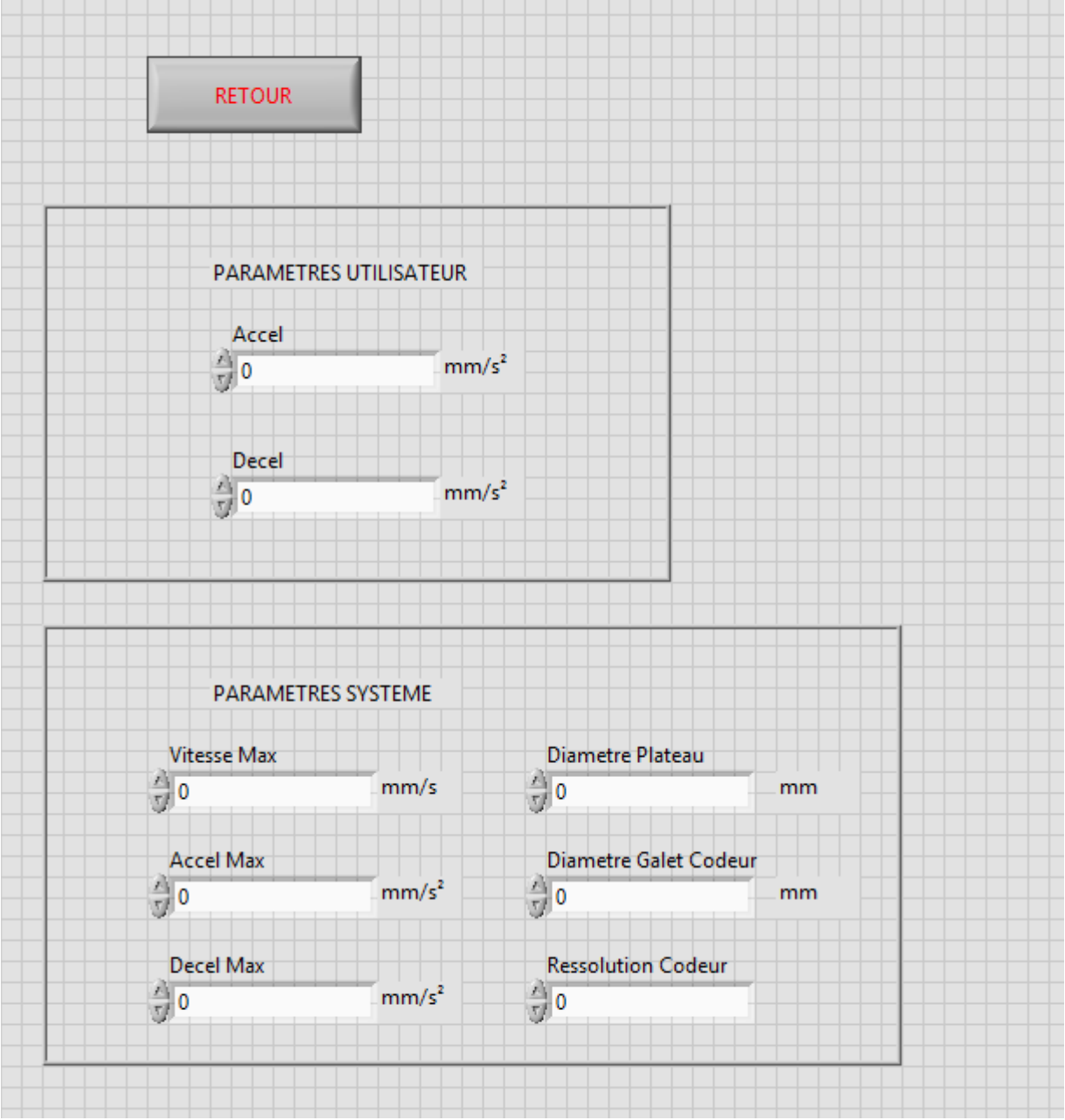
Chaque bouton Go, ici en bas de l'image, a un événement associé dans la structure et prend les valeurs d'angle et de vitesse associés dans l'interface pour lancer le VI Move.



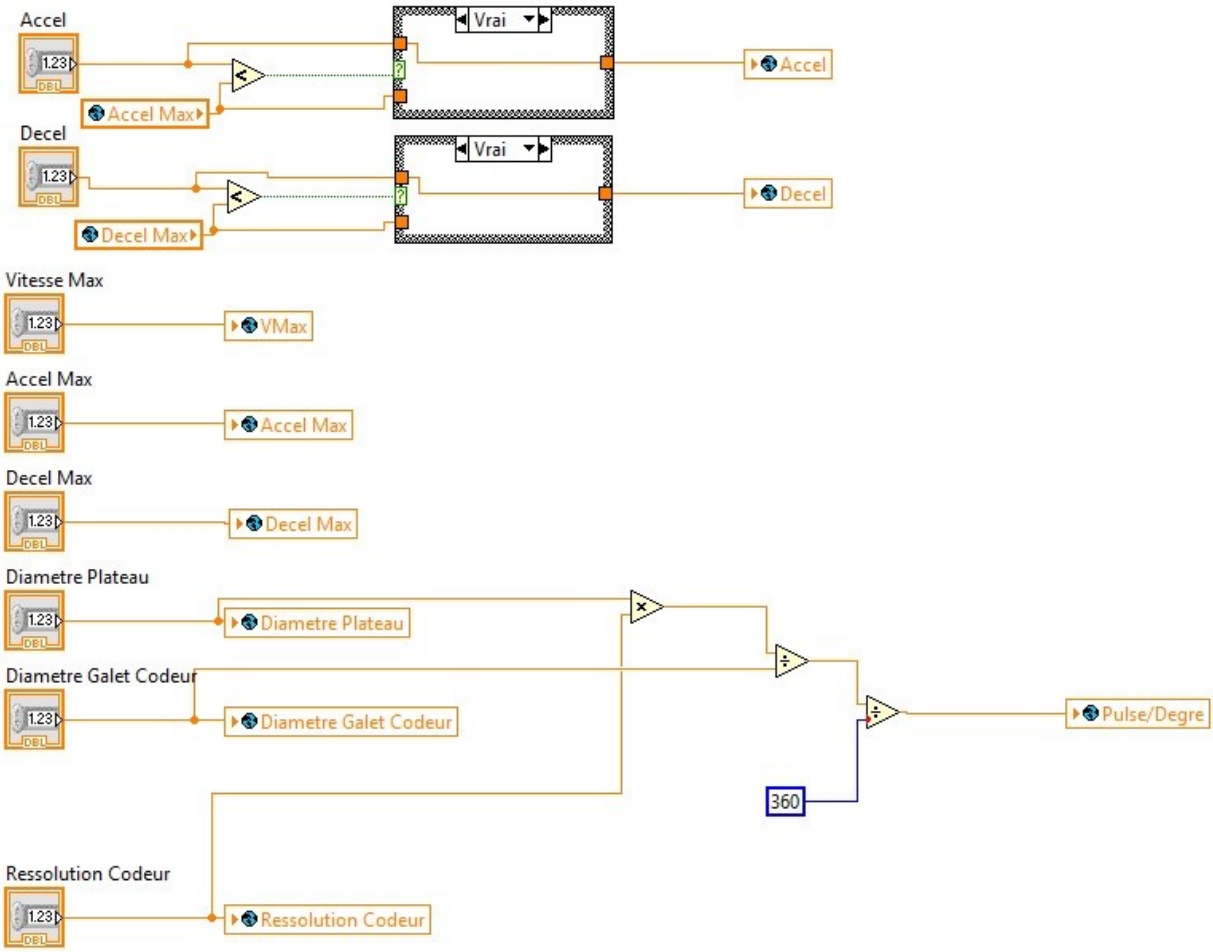
Toutes les parties du diagramme exposées à l'exception du déverrouillage et de la connexion sont dans une boucle while afin d'être constamment exécutables.

VI Param

Le VI Param est le VI appelé par le bouton Param System.



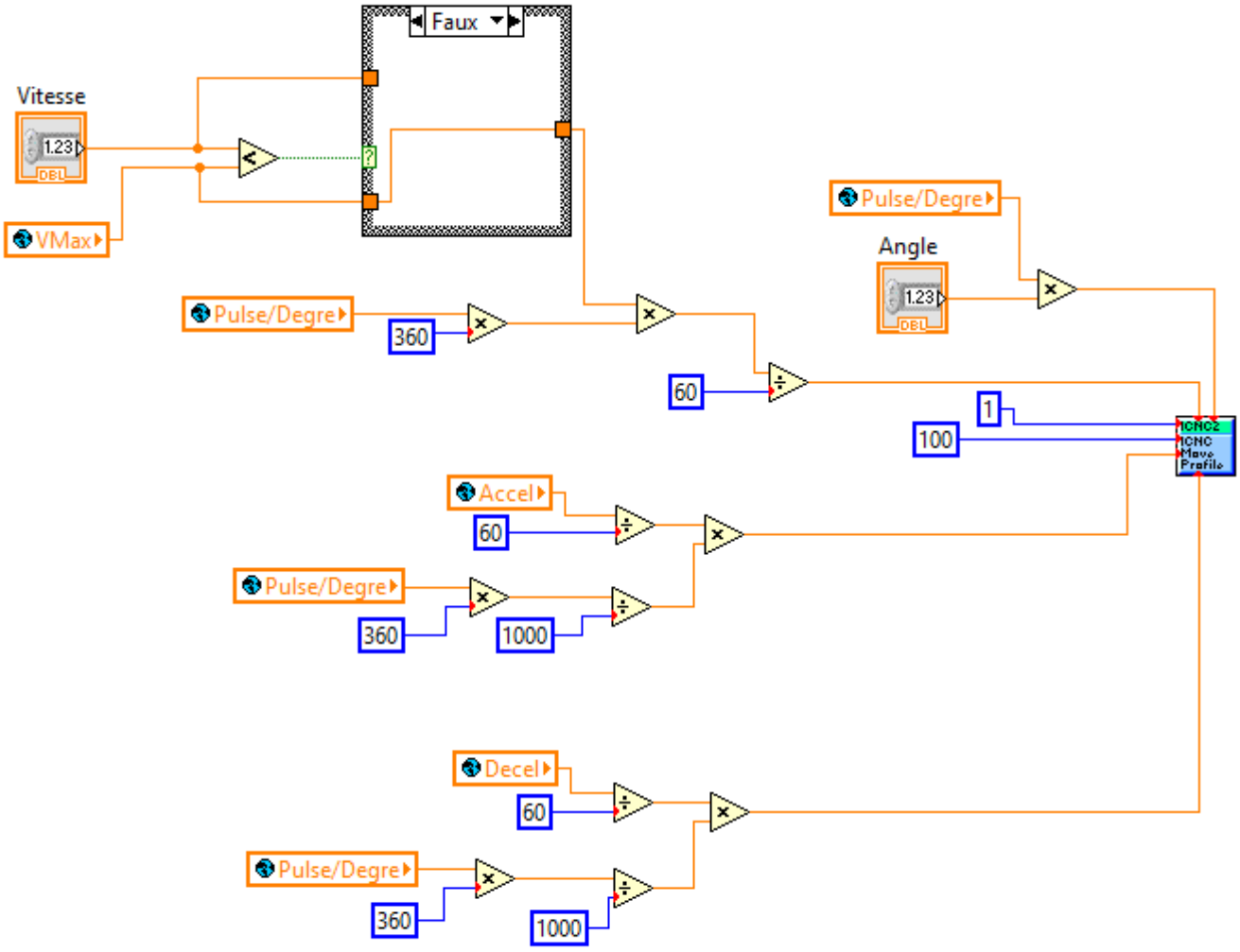
Ce VI permet de configurer les différents paramètres liés au système physique de la tournette et du matériel utilisé sur celle-ci.



Voici le contenu du VI Paramètres, toutes les valeurs rentrées dans le champ sont envoyées dans la variable globale du projet. Le champ pulse/degrés est calculé, et l'accélération et la décélération sont rentrées par l'utilisateur et sont comparées aux valeurs maximales définies.

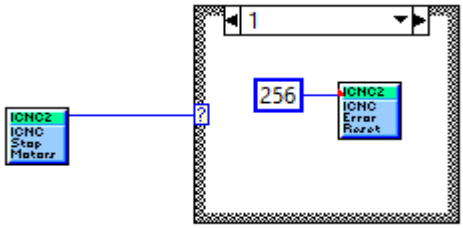
VI Move

Ce VI réalise les différentes conversions nécessaire à l'envoi de valeurs de type correct à la fonction MoveProfileAbs de la librairie ICNC. La vitesse est comparée à celle rentrée en vitesse maximum et les données sont formatées pour la fonction de la librairie. La vitesse, l'accélération et la décélération sont à fournir en khz/s².



VI Stop Moteurs

Enfin un VI stop moteurs existe car pour avoir un stop moteurs « instantané » nous utilisons la fonction StopMotorsAllAndClear(). Cependant, cette fonction verrouille aussi la carte. Nous avons fait ce VI pour lancer cette fonction et déverrouiller la carte juste après.



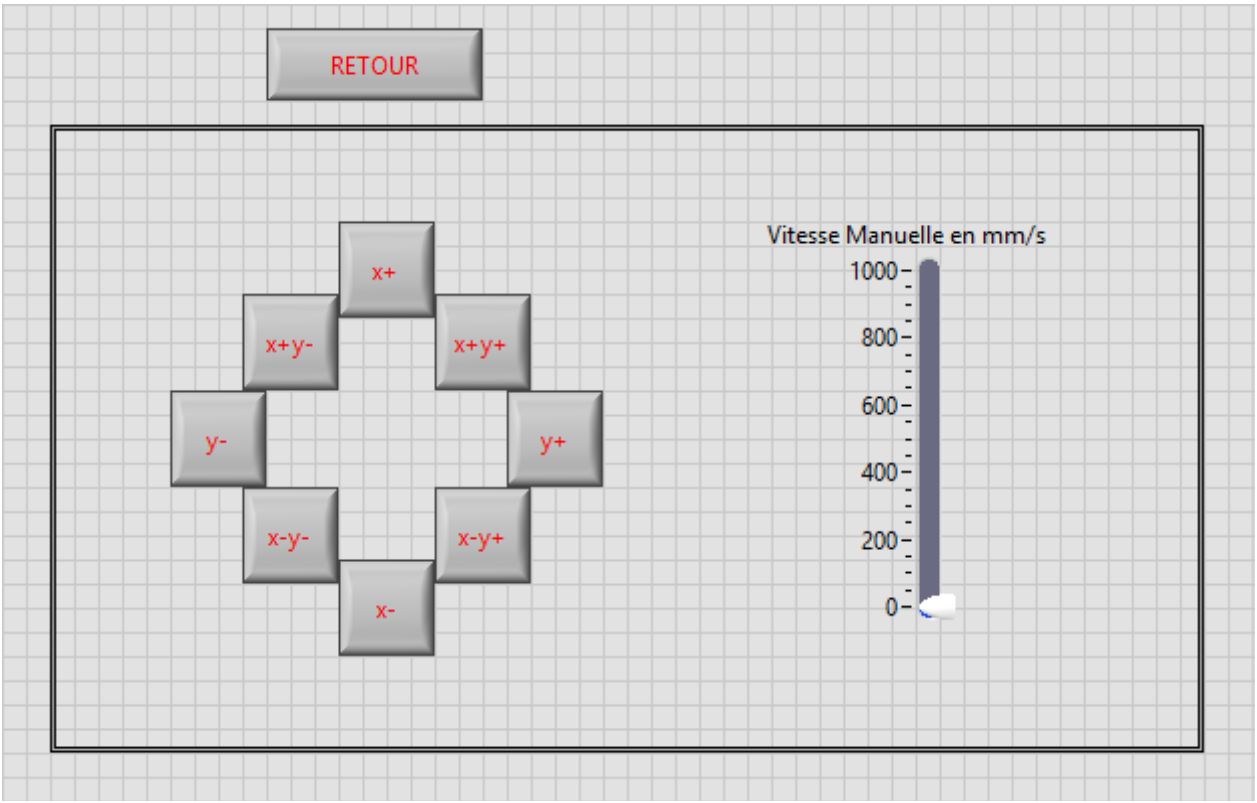
Exemple Commandes Manuelles Multiple-Axes

Un autre exemple a été créé pour montrer l'utilisation des fonctions de la librairie pour piloter manuellement 2 axes X et Y.

Plusieurs VIs ont été créés pour cet exemple, notamment un VI MoveAxe qui reprend le VI move précédemment défini avec une petite différence : l'axe n'est plus renseigné statiquement lors de l'appel à la fonction de déplacement, mais dynamiquement pour permettre la sélection de l'axe désiré pour la commande de déplacement.

VI Commandes Manuelles 2 Axes

Un VI Commandes Manuelles 2 Axes a été créé. Il repend le fonctionnement des boutons + et – de l'interface principale et l'applique aux axes X et Y.



Nous avons donc : un bouton x+, un bouton x-, un bouton y+, un bouton y-, un bouton x+y+, un bouton x+y-, un bouton y-x+, un bouton y+x- et un bouton x-y-.

Ces boutons permettent d'exécuter des déplacements sur les axes X et Y.
Ces boutons ont le même fonctionnement et les même consignes de sécurité à respecter que les boutons + et – de l'interface principale, c'est à dire, qu'ils doivent lancer le déplacement lorsqu'on

appuie sur le bouton et que le déplacement doit s'arrêter lorsqu'on lâche le bouton.
A cet effet, l'état des boutons est récupéré dans une variable globale afin que les commandes de déplacements et de stop soient lancées de façon unique à chaque utilisation d'un bouton et que ceux-ci ne soient pas envoyés en même temps à la carte. On test donc dans des boucles condition
un changement d'état et s'il y a eu changement d'état, on lance un déplacement si l'état est devenu vrai ou un stop si l'état est devenu faux.

Une glissière permet de définir la vitesse des déplacements et les paramètres systèmes sont ceux définis par l'utilisateur dans la fenêtre Param Système.

Ceci est reproduit pour chaque bouton dans une boucle while.