# *Tutoriel*

# *La commande de moteurs pas à pas à l'aide d'outils de saisie de données sur PC*

*par N. Shinjo, L. Buist et C. S. Subramanian*
*Florida Institute of Technology, Melbourne, Floride*

The use of LabVIEW is on the increase in universities and industries especially for data acquisition and process control. Frequently, there is a need to develop LabVIEW application software to work with foreign hardware, like a Slo-Syn stepper motor. Unfortunately, simple and clear procedures to develop motion control routines are not easily available. This paper describes a method developed to control a stepper motor using the LabVIEW 5.0 software and the National Instrument SCXI data acquisition system.

LabVIEW est de plus en plus utilisé dans les mondes universitaire et industriel, particulièrement pour la saisie de données et le contrôle de procédés. Régulièrement, des applications LabVIEW doivent être développées pour travailler avec du matériel étranger tel un moteur pas-à-pas Slo-Syn. Malheureusement, il n'existe pas de procédures simples et claires pour développer ces routines de gestion de mouvement. Cet article décrit une méthode pour contrôler un moteur pas-à-pas en utilisant le logiciel LabVIEW version 5.0 et le système de saisie de données SCXI de National Instrument.

## *1.0 Introduction*

Il arrive souvent que les processus de saisie de données impliquent divers types de commandes d'appareils. Cependant, les logiciels de saisie de données les plus courants n'incluent pas de pilotes de commande d'appareils ou de dispositifs en option qui soient polyvalents. L'utilisateur doit alors mettre au point l'interface de commande en fonction de l'application qu'il veut réaliser. Cet article décrit deux méthodes mises au point à la Florida Institute of Technology permettant de piloter un moteur pas à pas en employant la trousse de saisie de données très connue LabVIEW 5.0 et un port parallèle ou un port d'E/S numériques courant d'une carte de saisie de données. La première méthode utilise un port d'E/S numériques d'un module de saisie de données SCXI-1200 de la société National Instrument. L'autre méthode emploie le même logiciel, mais en compilant le code source externe (en C++), et pilote le moteur à partir d'un port parallèle. On peut utiliser le code source employé dans la deuxième méthode pour piloter le moteur de façon autonome, sans passer ni par une carte ni par un logiciel de saisie de données. Avec l'aide du module de conversion décrit dans cet article, on peut commander la majorité des moteurs pas à pas d'usage courant avec seulement deux fils de signaux numériques et un fil pour la masse.

On peut utiliser tout langage de programmation, tel le BASIC ou le C, ou un logiciel de saisie de données, avec des E/S numériques, pour piloter le moteur.

## *2.0 Système de commande de positionnement*

La figure 1 présente le schéma général du système de commande de positionnement. Celui-ci comprend un port de communication numérique à 8 bits, un pilote ou module de conversion, ainsi qu'un moteur pas à pas. On peut utiliser un port parallèle (ou port d'imprimante) ou bien des ports d'E/S numériques sur la carte de saisie de données pour piloter le moteur pas à pas par l'entremise du module de conversion. On dispose de deux méthodes de communication puisque les cartes externes de saisie de données typiques sont interfacées à l'ordinateur via le port parallèle.

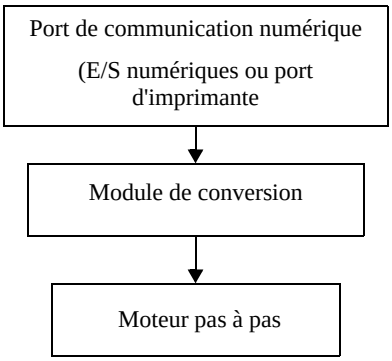



**Figure 1: Schéma général de la commande de positionnement**

Deux fils de signaux numériques et un fil de masse sont branchés au module de conversion. Un des fils de signaux numériques commande l'avancement et l'autre, le sens de rotation. Le module de conversion capte le signal en provenance du port de communication et le convertit en un séquencement de signaux alimentant les bobines du moteur pour le faire avancer. Le signal à la sortie du module de conversion est acheminé à chacune des bobines (un moteur typique compte deux paires de bobines). Les sections qui suivent décrivent en détail le câblage, les signaux et le code du programme.

### 2.1 Port parallèle de communication

On positionne le moteur pas à pas en injectant à l'entrée du module de conversion un signal de données à 8 bits. Pour acheminer les données à 8 bits à un port déterminé, il faut spécifier l'adresse du port et un code hexadécimal à 8 bits. Le tableau 1 présente les adresses des états des données du port parallèle (ou port d'imprimante) d'un PC.

**Tableau 1: Adresses des états des données**

| No du port | Données (en hexa) | État (en hexa) | Command (en hexa) |
|---|---|---|---|
| LPT1 | 378 | 379 | 37A |
| LPT2 | 278 | 279 | 27A |

On a utilisé les broches no2 (data 1), no9 (data 8) et no20 (masse) du port LPT1 (378 hexa) pour envoyer les signaux au module de conversion. La broche no2 commande l'avancement du moteur et la broche no9 détermine son sens de rotation. On doit noter que n'importe quelle combinaison de deux broches entre la no2 et la no9 peut servir à commander le moteur. Le tableau 2 donne un exemple de la désignation des broches du port parallèle. On emploie ici la séquence suivante.

- § Les états 1 et 0 (la broche no2 au niveau haut et la broche no9 au niveau bas) initialisent un pas en avant (c'est-à-dire dans le sens horaire).

- § Les états 1 et 1 (les broches no2 et no9 au niveau haut) initialisent un pas en arrière (dans le sens anti-horaire).
- § Les états 0 et 0 (les broches no2 et no9 au niveau bas) suivis par les états 1-0 ou 1-0 exécutent le pas.

Ainsi les deux instructions suivantes envoient les signaux au port LPT1 et font avancer le moteur d'un pas (typiquement 1,8 °). Notons que la commande " '_outp( )' " n'est valide qu'en Visual C++ de Microsoft. Les autres compilateurs peuvent employer une commande différente.

```
_outp(0x378, 1);   représentation binaire de 1 = 00000001
_outp(0x378, 0);   représentation binaire de 0 = 00000000
```

La première instruction met le bit 1 au niveau haut, les autres bits (du bit 2 au bit 8) étant au niveau bas. La seconde instruction met tous les bits à zéro.

La séquence de ces deux instructions force la bascule à changer d'état, ce qui fait avancer le moteur d'un pas.

De la même façon, la séquence des deux instructions suivantes actionne le moteur d'un pas dans le sens inverse (anti-horaire). La première instruction inverse le sens de rotation et la deuxième active la bascule et achemine un signal au moteur.

```
_outp(0x378, 255);représentation binaire de 255  =1111111
_outp(0x378, 0);représentation binaire de 0      =00000000
```

### 2.2 Le module de conversion

Le module de conversion génère des impulsions au moment où il reçoit les données de commande en provenance de l'ordinateur. Le front descendant de l'impulsion d'entrée active le circuit et envoie les signaux aux deux paires de bobines du moteur pas à pas. La première série de signaux inverse la polarité de la première bobine et la deuxième série fait de même pour la deuxième bobine, ce qui fait tourner le moteur d'un pas (1,8 °). Lorsque le circuit est sous tension, il alimente les bobines même si le moteur ne tourne pas. La figure 2 nous présente le schéma du circuit du module de conversion.

**Tableau 2: Branchement et désignation des broches de l'interface parallèle à un PC d'IBM (Hall, 1991)**

| No de broche | Signal | Description du signal | Sens du signal |
|---|---|---|---|
| 1 | STROBE | Impulsion STROBE lecture des données | IN/OUT |
| 2 | DATA 1 | Ces signaux désignent respectivement l'information des bits 1 à 8 des données parallèles. Chaque signal est au niveau " haut " quand la donnée est un " 1 " logique et au niveau " bas " quand la donnée est un " 0 " logique. | IN/OUT |
| 3 | DATA 2 | | IN/OUT |
| 4 | DATA 3 | | IN/OUT |
| 5 | DATA 4 | | IN/OUT |
| 6 | DATA 5 | | IN/OUT |
| 7 | DATA 6 | | IN/OUT |
| 8 | DATA 7 | | IN/OUT |
| 9 | DATA 8 | | IN/OUT |
| 10 | ACKNLG | Un niveau " bas " signale que la donnée a été reçue et que l'imprimante est prête à recevoir d'autres données. | OUT |
| 11 | BUSY | Un niveau " haut " signale que l'imprimante n'est pas en mesure de recevoir des données. | OUT |
| 12 | PE | Un niveau " haut " signale qu'il n'y a plus de papier dans l'imprimante. | OUT |
| 13 | SLCT | Ce signal indique que l'imprimante est sélectionnée. | OUT |
| 14 | AOTO FEED XT | Quand ce signal est " bas ", l'imprimante fait un saut de ligne après impression. | IN |
| 15 | NC | non utilisé | - |
| 16 | 0 V | Niveau logique 0 | - |
| 17 | CHASIS GND | Masse de l'imprimante | - |
| 18 | NC | non utilisé | - |
| 19 | GND - Masse | Masse | - |
| 20 | Masse | | - |
| 21 | Masse | | - |
| 22 | Masse | | - |
| 23 | Masse | | - |
| 24 | Masse | | - |
| 25 | Masse | | - |

### 2.3 Le moteur pas à pas

Le moteur pas à pas est un mécanisme qui positionne des charges en fonctionnant par incréments (ou pas) de décalage angulaire. Cette rotation pas à pas est réalisée en commutant l'alimentation aux bobines de façon à ce que les phases du moteur soient activées selon une séquence déterminée. La figure 3 présente les connexions typiques d'un moteur pas à pas (deux paires de bobines).

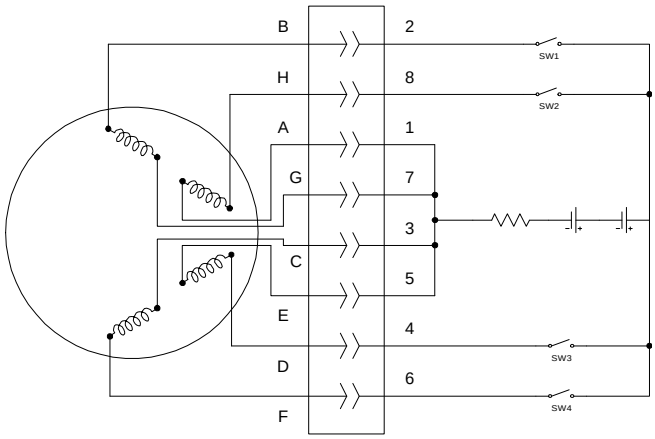



**Figure 3: Schéma des connections des bobines d'un moteur pas à pas**

Le tableau 3 nous indique la séquence de commutation requise pour faire avancer le moteur d'un pas. Le pas type est de 1,8 °, ce qui implique que 200 pas sont nécessaires pour faire un tour complet, soit une rotation de 360 °. Pour actionner le moteur dans le sens inverse, on intervertit la séquence de commutation (c'est-à-dire 4, 3, 2, 1).

**Tableau 3: Séquence de commutation des bobines**

| Pas | SW1 | SW2 | SW3 | SW4 |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 |

1 : niveau haut, 0 : niveau bas

Le moteur pas à pas peut également fonctionner en mode demi-pas (0,9 °). Ce mode nécessite cependant une séquence de commutation additionnelle et une circuiterie différente du module de conversion. De plus, ce mode réduit le couple de retenue du moteur. Dans le cadre de la présente application, seul l'incrément plein pas (1,8 °) est utilisé.

## *3.0 L'emploi du port parallèle d'un ordinateur pour la commande de positionnement*

La section 2.1 schématise la configuration de la commande de moteur par l'intermédiaire du port parallèle. En utilisant le module de conversion décrit à la section 2.2, on peut faire avancer le moteur d'un pas dans un sens spécifié en faisant exécuter les lignes de code source suivantes. Ce code est écrit en Visual C++ de Microsoft. Les commandes au port de communication parallèle varient selon le langage utilisé. Par exemple, en C++ de Borland, on a " outp " (adresse du port en hexa, code binaire) alors qu'en Visual Basic, c'est " out " (adresse du port en décimal, code binaire). On peut modifier la vitesse de rotation en jouant sur la période d'exécution de la boucle. Une commande plus précise de cette vitesse peut être réalisée par l'emploi d'une commande de minuterie (énoncé attente) au lieu d'une boucle.

```
do loop
_outp(0x378, 0 ou 255);
(une boucle vide servant à ajuster la période entre les commandes);
outp(0x378, 0);
(une boucle vide servant à ajuster la période entre les commandes);
(une boucle vide ou une minuterie servant à ajuster la période
d'exécution de la boucle);
end loop
```

La première commande " _outp " détermine le sens de rotation du moteur et la seconde, le fait avancer. À noter qu'avec un processeur plus rapide, il faudra peut-être utiliser des boucles vides (ou une commande de minuterie) afin de créer un temps d'attente entre les exécutions des commandes au moteur. Des exécutions trop rapides peuvent provoquer un positionnement irrégulier du moteur. Pour ce qui est du moteur utilisé ici, l'attente optimale entre les commandes était de 100 ms.

### 3.1 Positionnement à partir de LabVIEW

L'arrangement de base du positionnement par le port parallèle à l'aide de LabVIEW (figure 4) est le même que l'arrangement classique présenté à la section 2.1. On peut piloter le moteur avec un codage tout à fait identique à celui décrit à la section précédante. La seule différence est qu'au lieu de compiler le code source en un programme exécutable fonctionnant de façon autonome, ce code est compilé en un programme exécutable en LabVIEW. De cette façon, la routine de positionnement peut être utilisée conjointement avec des routines de saisie de données réalisées avec LabVIEW. À cette fin, la fonction LabVIEW appelée "

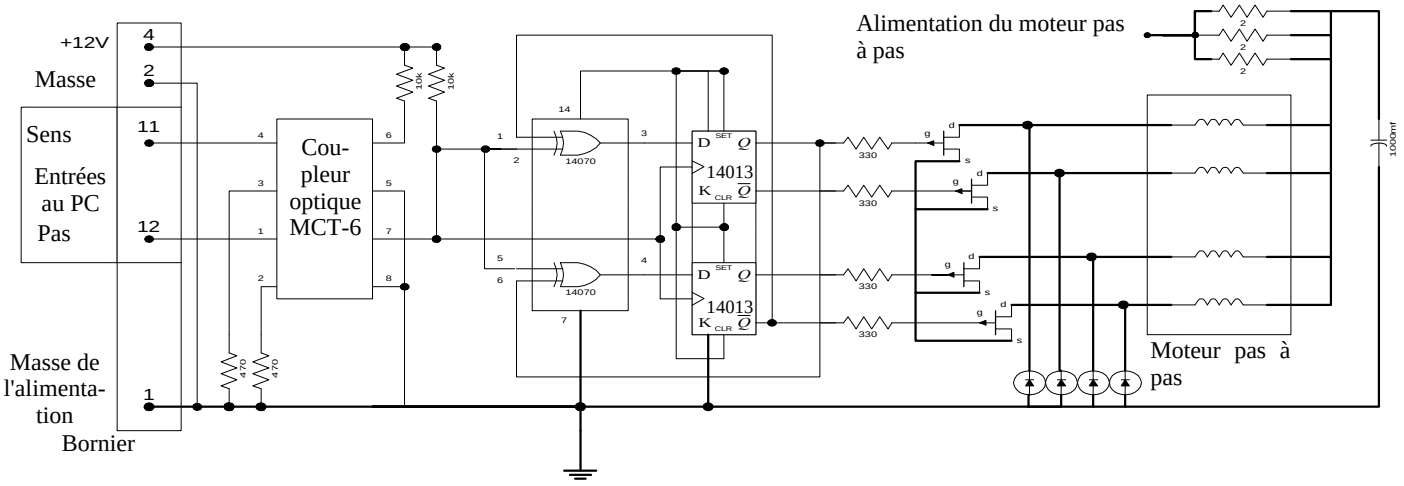



**Figure 2 : Schéma du circuit du module de conversion**

Code Interface Node (CIN) " (nœud d'interface avec le code) est employée pour lier, à destination de LabVIEW [3,4], le code externe écrit dans un langage de programmation classique. LabVIEW fait appel au code exécutable quand le nœud s'exécute, transférant les données d'entrée à partir du corps principal du programme vers le code exécutable, et retourne les données venant du code exécutable vers le corps principal. On peut compiler le code en opérations monothread ou multithread. Seuls les compilateurs suivants peuvent servir à créer un CIN.

- Le compilateur Visual C++ de Microsoft
- Le compilateur C de Symantec
- Le compilateur C/386 de Watcom dans l'environnement Windows 3.1

Le code externe en C++ chargé dans le CIN de LabVIEW est simplifié afin de minimiser les chances de blocage du programme lors de l'exécution du code du CIN. Ce code du CIN n'accomplit qu'un pas à la fois et retourne au VI de la commande de positionnement. Ce pas équivaut à une rotation de 1,8 °. Pour réaliser une série de pas, on appelle le CIN le nombre de fois qu'il faut. En insérant la boucle d'attente dans le VI de commande du positionnement, on peut régler la vitesse des pas suivants. L'inconvénient de cette méthode de commande est que la vitesse du moteur varie avec la vitesse du processeur. Cet état de chose peut être amélioré en plaçant la structure des boucles (selon de nombre de pas à exécuter) dans le code source en C.

### 3.1.1 Description des fichiers " make "

Les compilateurs ont besoin d'instructions leur précisant comment produire un projet ou un fichier. Ces instructions se présentent sous la forme de " make files " (fichiers " make "). LabVIEW, par exemple, installe un fichier " make " dans le répertoire LabVIEW\cintools qui renferme les instructions indiquant au compilateur comment produire un CIN générique (ntlvsb.mak); d'autres types de compilateur, cependant, auront besoin d'instructions supplémentaires pour savoir comment générer un CIN. Un fichier " make " spécifique, combiné à un fichier " make " LabVIEW générique, donnera l'information au compilateur lui permettant de produire un fichier .lsb (au lieu d'un fichier .exe). Le fichier .lsb sert à interfacer le code externe, comme le C, avec le programme LabVIEW. Le CIN n'accepte que les codes externes affectés de l'extension .lsb (il génère bien un fichier avec l'extension .c mais il refuse comme code source CIN un fichier avec l'extension .c). On peut trouver des renseignements supplémentaires à ce sujet dans la référence [2] " LabVIEW Code Interface Reference Manual ".

### 3.1.2 Étapes pour créer un CIN

Les procédures nécessaires à la création d'un CIN varient selon l'environnement et le compilateur utilisés. Dans cette section-ci, on illustre les procédures à utiliser avec le compilateur Microsoft Visual C++ sous Windows 95 de Microsoft. Des détails supplémentaires et des procédures touchant d'autres types de compilateurs se trouvent dans le " LabVIEW Code Interface Reference Manual ".

1. Avant de créer le CIN, déterminer quelles sont les données et le format de celles-ci (c'est-à-dire entier, point flottant, chaîne de caractères, etc.), à envoyer au/recevoir du CIN.

2. Écrire un code en C et s'assurer que celui-ci ne renferme aucune erreur.

3. À partir du schéma du VI LabVIEW, sélectionner une icône de CIN (Functions => Advanced) et la positionner sur le schéma. Elle devrait être similaire à la figure 5.

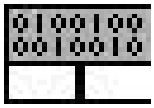


**Figure 5 : Icône de CIN**

4. Au départ, le CIN a une seule paire de bornes (un terminal d'entrée et un de sortie). On peut ajouter des bornes en faisant glisser avec la souris le coin inférieur gauche ou droit de l'icône ou en sélectionnant la fonction " Add Parameter " à partir du menu déroulant de la borne CIN (en cliquant avec le bouton droit sur les bornes de paramètres) (figure 6).

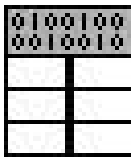


**Figure 6 : Icône de CIN avec entrées et sorties multiples**

5. Brancher les bornes aux panneaux de commande et de voyants (figure 7).

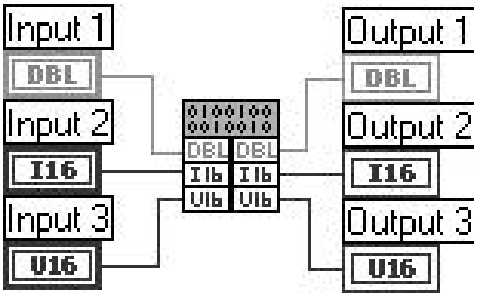



**Figure 7 : Icône de CIN avec ses bornes**

6. Sélectionner " Create .C File " à partir du menu déroulant (cliquer sur l'icône avec le bouton droit). Le nom que l'on veut donner au fichier sera alors demandé (filename.c format). Taper le nom de fichier du code source et l'enregistrer. Ce fichier enregistré renferme le code en C suivant.

```
/*
 * Fichier source du CIN
 */

#include "extcode.h"

CIN MgErr CINRun(float64 *Input_1, int16 *Input_2,
uInt16 *Input_3);

CIN MgErr CINRun(float64 *Input_1, int16 *Input_2,
uInt16 *Input_3)
{
/* TAPER VOTRE CODE ICI */
return noErr;
}
```

Il faut prendre note que les étiquettes (c'est-à-dire Input 1, Output 1, etc.) et leur format de donnée (c'est-à-dire double DBL, entier I6, etc.) écrits sur les boutons de commande et sur les voyants deviennent les noms de variables et les format de données du code en C. De plus, l'énoncé " #include "extcode.h " doit être le premier fichier d'en-tête. Les fichiers d'en-tête additionnels tel " math.h " et " stdio.h " doivent suivre ce premier fichier

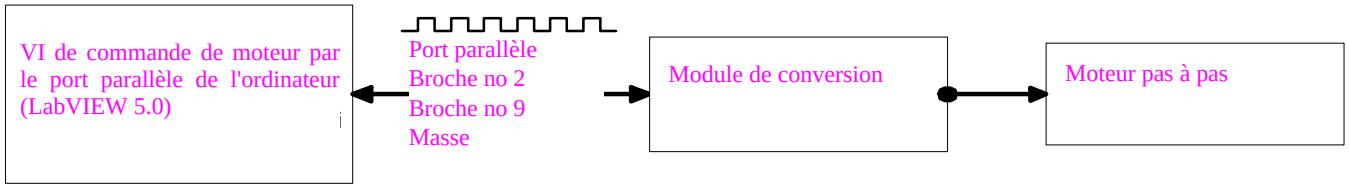



**Figure 4 : Schéma de l'emploi du port parallèle pour commander un moteur pas à pas**

7. Insérer le code en C que vous avez écrit dans le code en C généré par LabVIEW (où il est écrit /* TAPER VOTRE CODE ICI */). S'assurer que les noms et les types des variables concordent avec ceux des variables déclarées. Par exemple, nous aurions pour le cas présent :

Code en C pour la commande du moteur pas à pas

```
/*
 * Fichier source du CIN
 */

#include "extcode.h"
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

CIN MgErr CINRun(int16 *var1);

CIN MgErr CINRun(int16 *var1)
{
        {
        _outp(0x378, *var1);
        _outp(0x378, 0);
        }
        return noErr;
}
```

8. Créer un fichier " make " en utilisant un éditeur de texte (Notepad, Word, Wordpad, etc.) et l'enregistrer sous le même nom de fichier que le code source en C, mais avec l'extension .mak. Le fichier " make " doit renfermer les quatre lignes qui suivent.

```
Name = name of C file without extension
Type = CIN
CINTOOLSDIR = path to cintools directory
!include<$(CINTOOLSDIR)\ntlvsb.mak>
```

Par exemple, nous aurions dans le cas présent :

Fichier 'make' pour la commande du moteur pas à pas

```
name = step
type = CIN
cintoolsdir = c:\labview\cintools
!include <$(cintoolsdir)\ntlvsb.mak>
```

Le chemin d'accès vers le répertoire cintools se trouve (sous Windows 95) dans " C:\ProgramFiles\National Instruments\LabVIEW\Cintools ".

9. Avant de compiler le code source, s'assurer que le code source en C et le fichier " make " se trouvent dans le même répertoire (sinon le compilateur génèrera un ficher .exe au lieu d'un fichier .lsb).

10. Ouvrir le fichier " make " dans le Visual C++ de Microsoft. Quand le fichier est ouvert, le programme affiche deux boîtes de dialogue avec mise en garde. Le premier message dit : " Ce fichier " make " n'a pas été généré par Developer Studio ". À la question " Souhaitez-vous poursuivre? ", répondre " Oui ". Puis la seconde boîte de dialogue propose le type d'environnement dans lequel on veut fonctionner (le choix proposé par défaut est Win32). S'assurer que Win32 est sélectionné et cliquer sur "OK".

11. Sélectionner " Build filename.exe file " à partir du menu déroulant en Visual C++ de Microsoft. Le programme exécute les instructions contenues dans le fichier " make " et crée le fichier filename.lsb au lieu de filename.exe.

    Si le fichier .lsb est généré sans erreur, la fenêtre d'état émet le rapport suivant :

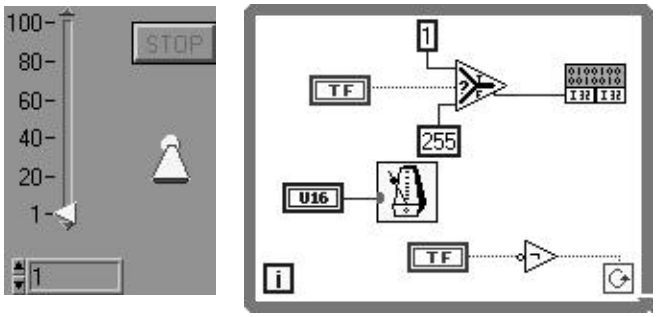
```
---------- Configuration: Filename - Win 32 Debug ----------
Microsoft (R) Program Maintenance Utility version 1.62.7022
Copyright (C) Microsoft Corp 1988-1997. All rights reserved.
filename.c
Microsoft (R) 32-Bit Incremental Linker Version 5.00.7022
Copyright (C) Microsoft Corp 1992-1997. All rights reserved.
LabVIEW resource file, type 'CIN', name 'filename.lsb', created properly.
Filename.exe - 0 erreur(s), 0 mise(s) en garde
```

    où filename est le nom du code source en C ou du fichier " make " compilé. Le fichier .lsb doit se trouver dans le répertoire où se trouve le code source en C et les fichiers " make ".

12. À partir du schéma du VI LabVIEW, (étape 3), cliquer avec le bouton droit sur l'icône du CIN et sélectionner " Load Code Resources " (Charger les ressources du code). La boîte de dialogue affiche le chemin d'accès aux fichiers .lsb que vous voulez charger. Sélectionner le fichier et cliquer sur " Open ".

13. Le code source est maintenant chargé à l'intérieur de l'icône du CIN et est prêt à exécuter la tâche. Une fois le code chargé, il deviendra une icône autonome et exécutable, c'est-à-dire qu'il n'aura pas besoin d'un code source en C ou d'un fichier " make ". Quand ce VI est enregistré (avec son icône de CIN) il peut être appelé à partir d'autres VI.

Un sous-VI LabVIEW, appelé Commande de Moteur.vi est maintenant prêt à piloter le moteur pas à pas par l'intermédiaire d'un port parallèle. La figure 8 nous présente son schéma du panneau de commande et son schéma-bloc.




STOP = ARRÊT

**Figure 8: Schéma du panneau de commande et schéma-bloc du sous-VI LabVIEW pour la commande de moteur pas à pas par l'intermédiaire d'un port parallèle.**

## *14.0 L'emploi du système SCXI pour la commande de positionnement*

On a utilisé le système Signal Conditoning Extension for Instrumentation (SCXI) de la société National Instrument avec LabVIEW 5.0 afin de mettre au point le programme permettant de réaliser une expérience de soufflerie impliquant un moteur pas à pas. Le système comprend un chassis SCXI-1000 muni de 4 fentes pour la saisie de données, le multiplexage et les modules de commande. Deux modules sont présentement installés dans le système : un SCXI-1200 et un SCXI-1121. Le SCXI-1200 est la carte de saisie de données d'usage général (carte A/N). Elle comprend quatre canaux différentiels (ou huit canaux à entrée simple) et trois ports d'E/S numériques à 8 bits. Le SCXI-1121 (avec son appareil auxiliaire SCXI-1321) compte quatre canaux différentiels isolés optiquement et ses circuits d'excitation. L'arrangement de base du système est le même que celui du contrôle de positionnement par le port parallèle. Étant donné que le système SCXI emploie déjà le port parallèle pour communiquer avec un ordinateur, on a utilisé le port d'E/S numériques du SCXI-1200 pour piloter le moteur pas à pas.

### 4.1 Branchement du moteur pas à pas (SCXI-1200)

Le port d'E/S numériques du SCXI (bornes PA-0, PA-7 et la masse des signaux numériques) pilote le moteur pas à pas, via le module de conversion, comme le montre la figure 9. On peut retrouver les descriptions détaillées des ports dans le manuel du SCXI-1200 [3]. La combinaison des impulsions numériques (décrite plus haut) et la fréquence des signaux en PA-0 et PA-7 déterminent le sens de rotation et la vitesse du moteur.

Le tableau 4 donne les détails du branchement entre le SCXI-1200 (50 broches) et le bornier du module de conversion. Pour le branchement entre le module de conversion et le moteur pas à pas, il faut se référer au schéma du circuit du module de conversion à la figure 2.

**Tableau 4: Brochage du connecteur (SCXI-1200)**

| No de broche du SCXI-1200 (50 broches) | Numéro de borne du module de conversion |
|---|---|
| 13 (masse des signaux numériques) | 3 (masse) |
| 14 (PA-0) | 11 (signal d'avancement) |
| 21 (PA-7) | 12 (signal du sens de rotation) |

Un sous VI LabVIEW, appelé DIO Motor Control (one step).vi (Commande de moteur par E/S numériques [pas unitaire]) a été conçu pour piloter un moteur pas à pas. La figure 10 nous présente son schéma de panneau de commande et son schéma-bloc.

## *5.0 Sommaire*

Cet article traite des méthodes de commande d'un moteur pas à pas par l'intermédiaire d'un port parallèle et du module SCXI de la société National Instrument. L'utilisation du module de conversion, qui y est décrit, permet de piloter un moteur pas à pas en se servant de langages de programmation ou de logiciels de saisie de données classiques. Il s'agit d'une façon économique d'élaborer un prototype de système de commande/saisie de données. De plus, le logiciel de saisie de données, qui est capable de compiler des codes de programmation classiques (tel le Code Interface Node [CIN] de LabVIEW), permet d'incorporer la routine de commande de positionnement dans la fonction de saisie de données. La procédure est expliquée ici en détail pour la routine de commande écrite en C++. On peut appliquer cette procédure à n'importe quelle routine de commande de positionnement, comme le déplacement d'une sonde, la rotation d'un modèle, etc.

## *6.0 Remerciements*

Les auteurs sont très reconnaissants à M. Dan Simpson de son assistance dans le laboratoire.

## *7.0 Références*

[1]. Hall, Douglas V., "Microprocessors and Interfacing: Programming and Hardware", McGraw-Hill, 2e éditon, 1991.

[2]. National Instruments, "LabView Code Interface Reference Manual", National Instruments, édition de janvier 1998, no de pièce 320539D-01.

[3]. National Instruments, "LabView User Manuel", National Instruments, édition de janvier 1998, no de pièce 320999D-01.

[4]. National Instruments, "C Programming Reference Manual", National Instruments, édition de janvier 1998, no de pièce 321296B-01.

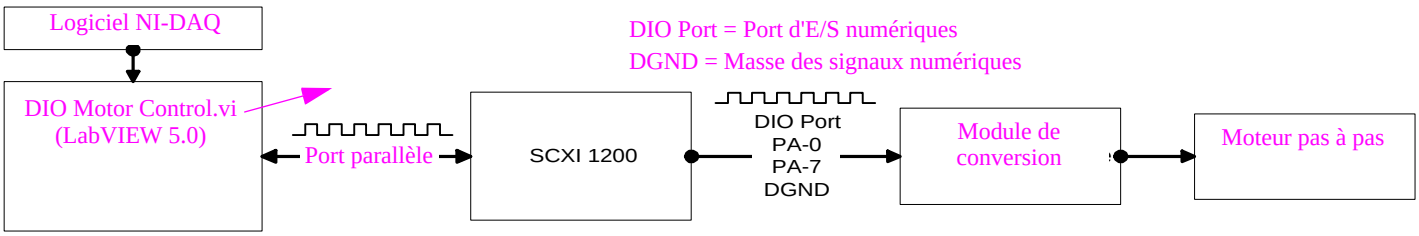
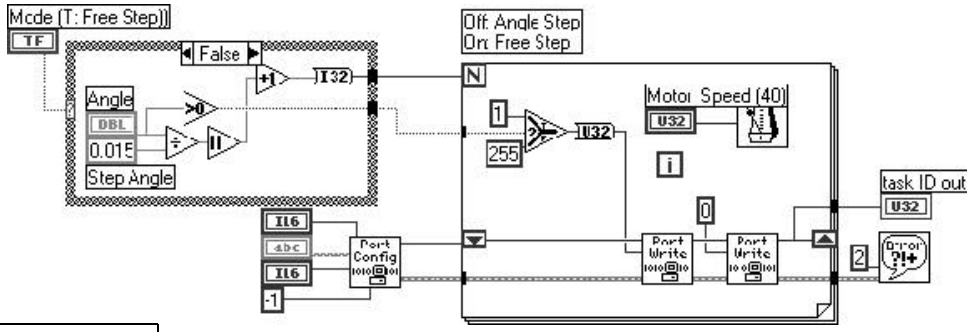



**Figure 9: Schéma montrant l'emploi du SCXI-1200 pour commander un moteur pas à pas**

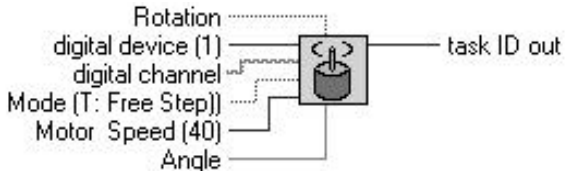
# DIO Motor Control (one step).vi

## Schéma-bloc

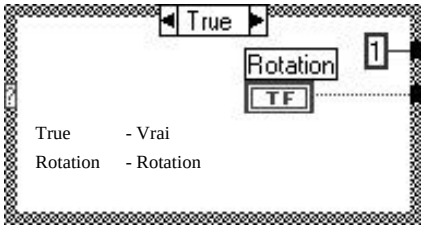
| | |
|---|---|
| task ID out | - sortie d'identification de la tâche |
| Mode (T: Free Step) | - Mode (T: avancement libre) |
| False | - Faux |
| Off: Angle Step | - Hors: Avancement angulaire |
| On: Free Step) | - En: Avancement libre) |
| Angle | - Angle |
| Step Angle | - Angle du pas |
| Motor Speed (40) | - Vitesse du moteur (40) |
| task ID out | - sortie d'identification de la tâche |
| Port Config | - Config. du port |
| Port Write | - Écrit. port |




## Panneau des connexions




| | |
|---|---|
| digital device (1) | = appareil numérique (1) |
| digital channel | = canal numérique |
| Mode (T: Free Step) | = Mode (T: avancement libre) |
| Motor Speed (40) | = Vitesse du moteur (40) |
| task ID out | = sortie d'identification de la |




## Panneau de commande

| | |
|---|---|
| digital device | = dispositif numérique |
| Motor Speed (40) | = Vitesse du moteur (40) |
| Rotation | = Rotation |
| digital channel | = canal numérique |
| task ID out | = sortie d'identification de la tâche |
| port width (8) | = Nbre de bits du port |
| Mode (T: Free Step) | = Mode (T: avancement libre) |
| Angle | = Angle |
| On: Free Step | = En: avancement libre |

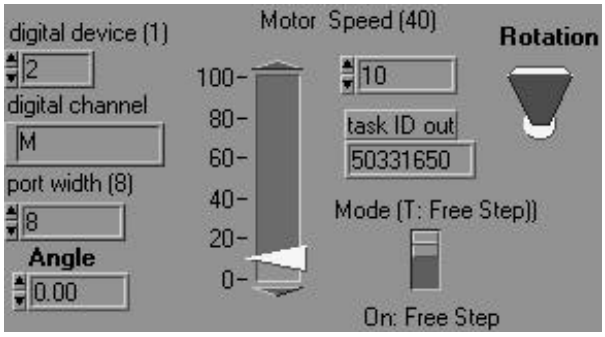



**Figure 10: Panneau de commande et schéma-bloc de la sous-VI LabVIEW pour la commande par SCXI de moteur pas à pas**

## *À propos des auteurs*


**Nagahiko Shinjo** poursuit ses études de doctorat en génie océanique à la Florida Institute of Technology. Il a obtenu sa M.S., également en génie océanique, de la même université. Ses domaines principaux de recherche comprennent le contrôle de la bioadhésion, les systèmes de commande de véhicules sous-marins autonomes et les appareils de commande et mesure submersibles. Il met présentement au point un système de ballast à alliage à mémoire de forme destiné aux véhicules sous-marins autonomes ou aux véhicules à base biologique.

Courriel : shinjon@winnie.fit.edu

**Larry Buist**, qui travaille depuis 13 ans à la Florida Institute of Technology, possède une expérience considérable dans la création de prototypes en collaboration avec les sociétés Harris GISD (Intersil), MicroPac Industries et plusieurs jeunes entreprises en électronique au Nevada, au Texas et en Floride. Il détient un brevet dans l'industrie des jeux vidéo et œuvre dans la conception et la fabrication de circuits électroniques et électromécaniques au sein de divers groupes de recherche à la Florida Institute of Technology.

Courriel : lbuist@fit.edu

**Chelakara S. Subramanian** est professeur agrégé en génie aérospatial à la Florida Institute of Technology. Il a obtenu son Ph.D. en génie mécanique de l'université de Newcastle en Australie et une M.E. en génie aérospatial de l'Indian Institute of Science en Inde. Ses compétences de recherche résident dans la mécanique des fluides expérimentale et dans les nouveaux appareils de commande et de mesure. Il travaille présentement sur les projets de recherche suivants : systèmes d'application de peinture sensibles à la pression et à la température; vélocimétrie par imagerie des particules; système économique de vélocimétrie Doppler par laser à deux composants et fonctionnant à diode; mesure sur le terrain des débits dans les ouragans; études en génie des vents. Il est fellow associé de L'AIAA, membre de l'ASME, membre de la Society of Engineers de Grande-Bretagne, ingénieur autorisé en Grande Bretagne et membre de la Société internationale des ingénieurs de France.

Courriel : subraman@winnie.fit.edu