Conception, réalisation et utilisation d'une interface PC pour codeurs incrémentaux

Dominique JACOB[1], Patrick LAGONOTTE[2]
I.U.T. de Poitiers, 6 allée Jean MONNET
86010 Poitiers Cedex
[1] Département Génie Electrique et Informatique Industrielle, site de Poitiers dominique.jacob@univ-poitiers.fr
[2] Département Hygiène, Sécurité, Environnement, site de Niort lagonotte@let.ensma.fr

RESUME : Pour disposer d'une mesure de position par un codeur incrémental, nous faisons réaliser par les étudiants une interface d'acquisition utilisant le port parallèle d'un PC. Le décodage de la position à partir des signaux bruts du codeur incrémental est réalisé par un circuit programmable (CPLD). Celui-ci effectue un décodage de quadrature à l'aide d'une machine synchrone et permet d'incrémenter ou de décrémenter le compteur de position. Le circuit programmable CPLD effectuant cette opération est configuré par un langage de description matériel (VHDL). Le port parallèle du PC est programmé avec une acquisition multiplexée de manière de pouvoir lire un mot de 16 bits à partir d'un registre 8 bits. Une application à l'identification des paramètres d'un pendule oscillant est effectuée à l'aide de l'ensemble, le PC étant programmé en langage C. La réalisation d'un prototype semble indispensable pour finaliser un travail technique, les seules conceptions logicielles ne permettant pas d'appréhender les réelles difficultés liées à sa concrétisation.

Mots clés : codeur incrémental, décodeur de quadrature, VHDL, circuit programmable CPLD, Langage C.

1 INTRODUCTION

La définition en modules, des programmes pédagogiques, conduit à une séparation artificielle des connaissances. Il importe de conduire des activités qui permettent de rassembler l'ensemble des compétences acquises. Dans cet esprit, on propose la réalisation et l'utilisation d'une interface entre un codeur incrémental et le port parallèle d'un PC. La carte électronique est à réaliser en utilisant un logiciel de CAO comporte un CPLD qui doit être configuré à l'aide du langage VHDL. Le PC, quant à lui, est programmé en C et une application d'identification est demandée. Ce travail rassemble ainsi quatre domaines de connaissances (CAO en électronique, VHDL, C, automatique) qui sont en général enseignés dans des modules distincts. Ce travail est exploité en DUT Dénie Electrique et Informatique Industrielle en 28 heures. A ce niveau et compte tenu du temps imparti, de nombreux problèmes ne sont pas abordés. Par exemple on ne prend pas en compte l'influence du système d'exploitation du PC sur l'échantillonnage, on se limite à l'identification d'un pendule autour de sa position stable, et le système d'acquisition est considéré comme très rapide devant le processus et on fixe d'emblée la résolution à 16 bits. Le but de ce travail est de rassembler différents problèmes techniques dans un même projet mais non de présenter une solution industrielle.

2 CONCEPTION DE L'INTERFACE

On utilise un codeur incrémental pour mesurer la position angulaire d'un pendule simple et identifier ses paramètres, c'est à dire sa pulsation propre et son coefficient d'amortissement par un simple essai de lâcher. Le dispositif est présenté figure 1.

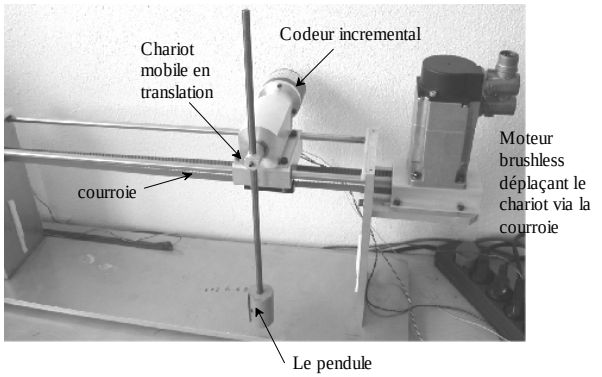


Fig. 1 : Le pendule étudié

Le pendule est mobile sur un chariot dont la position est commandée par un moteur brushless. La position angulaire du pendule est mesurée par un second codeur incrémental.

2.1 Schéma de principe

On prévoit une interface pour deux codeurs incrémentaux. Selon la taille du circuit programmable on pourra même augmenter ce nombre. L'application présentée ici ne comporte qu'un seul codeur.

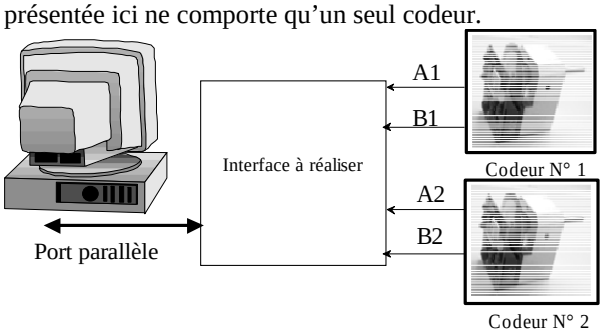


Fig. 2 : L'interface à réaliser

Chaque codeur comporte 500 points par tours, et délivre deux signaux A, B en quadrature (voir annexe). Le décodage de quadrature au quart de pas conduit à

Article available at http://www.j3ea.org or http://dx.doi.org/10.1051/j3ea:2008003

500x4 = 2000 points par tour. Cela nécessite un comptage sur au moins 11 bits (2^{11} = 2048). En pratique on adopte un comptage sur 16 bits (jusqu'à 2^{16}-1 = 65535) ainsi on pourra mesurer des positions correspondant à $32 = \frac{2^{16}}{2^{11}} = 2^5$ tours de codeur environ.

2.2 Rappel sur le décodage de quadrature

Le décodage de quadrature est obtenu par une machine d'état synchrone dont le diagramme est rappelé figure 3. Il faut disposer d'une horloge *CLK* dont la fréquence est toujours plus élevée que celle des signaux *A* et *B*. L'état des signaux *A* et *B* est pris en compte à chaque front montant de l'horloge *CLK*. Ces deux signaux permettent de coder 4 états *S1* (si *A*=0 et *B*=0), *S2* (si *A*=0 et *B*=1), *S3* (si *A*=1 et *B*=1), *S4* (si *A*=1 et *B*=0) et c'est l'ordre de passage d'un état à l'autre qui indique le sens de rotation.

Pour un sens de déplacement on obtient la séquence *S1* $\rightarrow$ *S2* $\rightarrow$ *S3* $\rightarrow$ *S4* $\rightarrow$ *S1* et à chaque transition il faut décrémenter une variable de comptage *N*. Pour l'autre sens de déplacement la séquence sera *S1* $\rightarrow$ *S4* $\rightarrow$ *S3* $\rightarrow$ *S2* $\rightarrow$ *S1* et il faut incrémenter la même variable de comptage N.

A cette description il faut ajouter la remise à 0 de la variable de comptage et l'initialisation de l'état en fonction de A et B, par exemple état *S1* si *A*=0 et *B*=0.

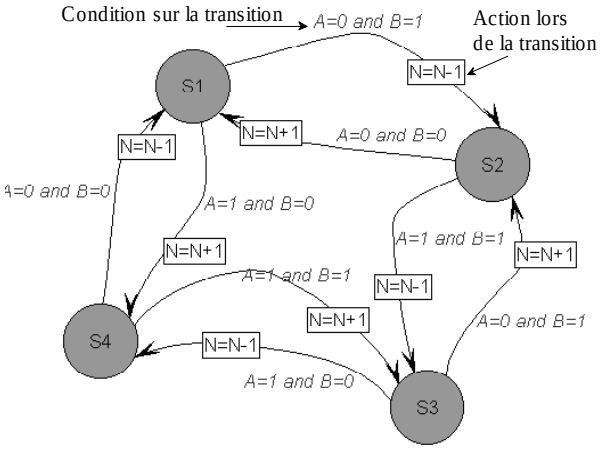


Fig. 3 : Graphe d'état pour la détection du quart de pas

2.3 Le port parallèle du PC

Les signaux disponibles et leur utilisation sont décrits de manière exhaustive dans la référence [2]. On présente tableaux 1 et 2, un résumé. Physiquement l'interface est un connecteur DB25 et on utilisera le port LPT1 dont l'adresse de base est 0x378. L'état des signaux est alors contrôlé par le contenu des registres d'adresses 0x378, 0x379 et 0x37A.

Ici on n'utilise que les deux registres 0x378 et 0x37A dont l'utilisation des signaux est rappelée dans les tableaux 1 et 2.

N° bit	Nom	N° Pin	Direction	inversé sur le connecteur
0	D0	2	In-Out	Non
1	D1	3	In-Out	Non
2	D2	4	In-Out	Non
3	D3	5	In-Out	Non
4	D4	6	In-Out	Non
5	D5	7	In-Out	Non
6	D6	8	In-Out	Non
7	D7	9	In-Out	Non

Tableau 1 : Registre d'adresse 0x378 (Donnée)

Le registre des données ne comporte que 8 bits et le comptage est prévu sur 16 bits cela nécessite un multiplexage par un bit noté MSB. De plus il y a deux codeurs distingués par un bit N_Codeur. On a aussi besoin de 2 signaux de sortie, RAZ1, RAZ2, pour commander la remise à zéro du comptage pour chaque codeur. Ainsi les 4 bits du registre de commande seront utilisés. On a choisi, Strobe=N_Codeur, AutoLF = RAZ1, Init = RAZ2, SelectIn = MSB.

On note N1 le résultat sur 16 bits du comptage du codeur N°1, les 8 bits de poids faibles de N1 sont notés LSB1 et les 8 bits de poids forts sont notés MSB1 (De même N2, MSB2, LSB2 pour le codeur N°2).

N° bit	Nom	N° Pin	Direction	inversé sur le connecteur
0	Strobe	1	Out	Oui
1	AutoLF	14	Out	Oui
2	Init	16	Out	Non
3	SelectIn	17	Out	Oui
4	EnIrq	Non connecté		
5	Dir	Non connecté		
6	Inutilisé	Non connecté		
7	Inutilisé	Non connecté		

Dir = 1 ==> 0x378 en entrée

Dir = 0 ==> 0x378 en sortie

Tableau 2 : Registre d'adresse 0x37A (commande)

Les 4 octets LSB1, MSB1, LSB2, MSB2 seront présents sur le registre de donnée en fonction des bits de commande N_Codeur et MSB selon le tableau 3.

N_Codeur	MSB	Registre de donnée
0	0	LSB1
0	1	MSB1
1	0	LSB2
1	1	MSB2

Tableau 3 : Adressage des registres

2.4 Le circuit programmable (CPLD)

On a utilisé trois circuits, un CPLD cypress CY37064 comportant 64 macrocellules et d'autre part un CPLD Altera Max7128S comportant 128 macrocellules et un FPGA Altera Flex 10k70 comportant 10 000 portes. Ces deux derniers circuits sont présents sur un kit de

développement (University program Design Laboratory Package) ce qui simplifie la mise en œuvre. Pour chacun d'eux la programmation du circuit en VHDL est identique seule change l'affectation des signaux d'entrée-sorties.

2.5 Description VHDL du circuit

Les signaux d'entrées-sorties sont précisés dans l'entity. L'architecture comprend 3 process concurrents, deux pour le décodage de quadrature associé à chaque codeur, et un pour gérer le registre de donnée de 8 bits. Les signaux nécessaires sont donnés ci-dessous.

```
entity deux_codeurs_lpt16bits is
    port(
        clk : in std_logic;
        RAZ1 : in std_logic;    --bouton de remise à 0
        A1 : in std_logic;  --sortie A du codeur n°1
        B1 : in std_logic;  --sortie B du codeur n°1
        RAZ2 : in std_logic;    --bouton de remise à 0
        A2 : in std_logic;  --sortie A du codeur n°2
        B2 : in std_logic;  --sortie B du codeur n°2
        N_codeur: in std_logic; -- numero du codeur, 0 pour codeur1, 1 pour codeur 2
        MSB: in std_logic; -- Sortie <= MSB si 1 sinon Sortie <=LSB
        Sortie : out std_logic_vector (7 downto 0)
        );
end deux_codeurs_lpt16bits;
```

L'entity décrivant les signaux d'entrée-sortie

```
type etat_codeur is (S1, S2, S3, S4);

signal etat1: etat_codeur;
signal count1: std_logic_vector(15 downto 0);
alias MSB1 : std_logic_vector(7 downto 0) is count1( 15 downto 8);
alias LSB1 : std_logic_vector(7 downto 0) is count1( 7 downto 0);

signal etat2: etat_codeur;
signal count2: std_logic_vector(15 downto 0);
alias MSB2 : std_logic_vector(7 downto 0) is count2( 15 downto 8);
alias LSB2 : std_logic_vector(7 downto 0) is count2( 7 downto 0);
```

Les signaux internes

Le graphe d'état de la figure 3 conduit à la description VHDL suivante pour le comptage. On a utilisé deux fonctions INC et DEC pour réaliser l'incrémentation et la décrémentation.

```
function INC ( E : std_logic_vector(15 downto 0) )
          return std_logic_vector is
variable S : std_logic_vector(15 downto 0);

begin
  S := E+1;
  return S;
end INC;
```

```
function DEC ( E : std_logic_vector(15 downto 0) )
          return std_logic_vector  is
variable S : std_logic_vector(15 downto 0);

begin
  S := E-1;
  return S;
end DEC;
```

Deux fonctions auxiliaires

```
    G_sortie : process (N_codeur,MSB,LSB1,MSB1,LSB2,MSB2)
    -- gestion de la sortie multiplexée
    -- N_codeur    MSB    sortie
    --    0        0      LSB1
    --    0        1      MSB1
    --    1        0      LSB2
    --    1        1      MSB2
-------------------------------------------------------------
    begin
         if N_codeur = '0' then
                if MSB='0' then
                     Sortie <= LSB1;
                else
                     Sortie <= MSB1;
                end if;
         else
                if MSB='0' then
                     Sortie <= LSB2;
                else
                     Sortie <= MSB2;
                end if;
         end if;
    end process G_sortie;
```

Le process de gestion du registre de donnée

```
comptage1 : process (A1,B1,RAZ1,clk)
  variable N1 : std_logic_vector (15 downto 0);
  variable var_etat1 : etat_codeur;

  begin - RAZ et detection de l'etat initial
      if RAZ1 = '0'  then
              N1 := "0000000000000000";
              if A1='0' then
                    if B1='0'  then
                         var_etat1 := S1;
                        else
                          var_etat1 := S2;
                    end if;
              else
                    if B1='0'  then
                            var_etat1 := S4;
                        else
                            var_etat1 := S3;
                    end if;
              end if;
              etat1 <= var_etat1;

  elsif clk'event and clk='1' then
      case etat1 is
                    when S1 =>
                        if A1='1' and B1='0' then
                            etat1 <= S4;
                            N1 := INC(N1) --N1:=N1+1;
                        elsif A1='0' and B1='1' then
```

Cette description conduit au fonctionnement donné par les chronogrammes des figures 4 et 5 (obtenus par simulation). Sur la figure 4, B1 est en avance sur A1, la variable de comptage est décrémentée et mise à 0 quand RAZ1 = 0. La durée d'activité des états 0 et 2 est plus courte d'une période d'horloge que celle des états 1 et 3. En réalité la période d'horloge est bien plus courte que celle des signaux A1 et B1 et ceci est imperceptible. Comme MSB = 0 la sortie est égale au LSB de N1.

Un process de décodage de quadrature

```
                elsif A1='0' and B1='1' then
                    etat1 <= S2;
                    N1 := DEC(N1); --N1:=N1-1;
                end if;
            when S2 =>
                if A1='0' and B1='0' then
                etat1 <= S1;
                N1 := INC(N1); -- N1:=N1+1;
                elsif A1='1' and B1='1' then
                    etat1 <= S3;
                    N1 := DEC(N1);--N1:=N1-1;
                end if;
            when S3 =>
                if A1='1' and B1='0' then
                    etat1 <= S4;
                    N1 := DEC(N1);--N1:=N1-1;
                elsif A1='0' and B1='1' then
                    etat1 <= S2;
                    N1 := INC(N1);--N1:=N1+1;
                end if;
            when S4 =>
                if A1='1' and B1='1' then
                    etat1 <= S3;
                    N1 := INC(N1);--N1:=N1+1;
                elsif A1='0' and B1='0' then
                    etat1 <= S1;
                    N1 := DEC(N1);--N1:=N1-1
                end if; --when others => null; est inutile
        end case;
    end if;
    count1 <= N1;
end process comptage1;
```

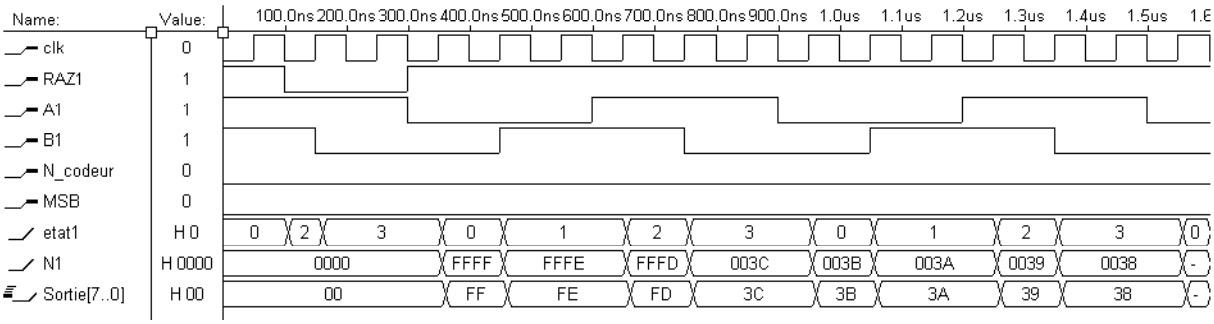


Fig. 4 : Signaux lors de la remise à zéro du comptage

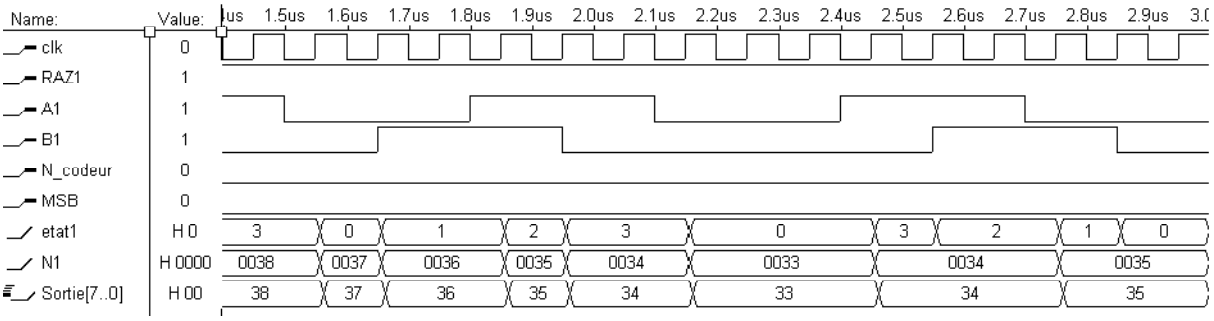


Fig. 5 : Signaux lors du changement de sens de rotation du codeur

Sur la figure 5, on observe les signaux lors du changement de sens de rotation du codeur. Au début B1 est en avance sur A1, la variable N1 décroît, puis pour t > 2,4 µs c'est A1 qui est en avance sur B1, alors N1 croît.

2.6 Le schéma électronique et la réalisation

L'interfaçage des codeurs incrémentaux alimentés en 15 V nécessite un adaptateur de tension à 5V, ici on utilise un 4049. Le signal d'horloge est obtenu par un quartz associé à une porte inverseuse du 4049. Afin de mettre en forme le signal d'horloge et de disposer de plusieurs fréquences possibles on a utilisé un diviseur de fréquence 4040. La protection des signaux issus du PC sur le port parallèle est assurée par des buffers 74244. La figure 6 montre la réalisation obtenue.

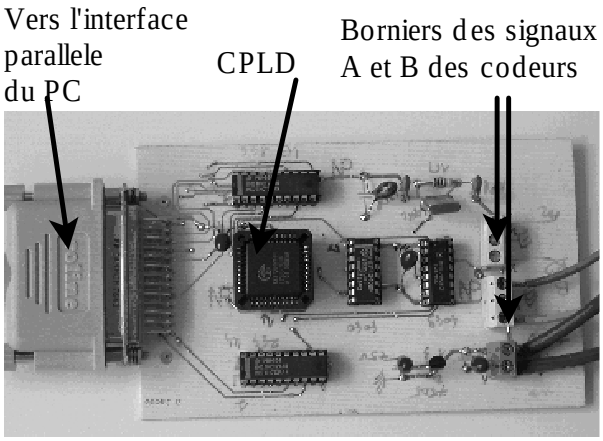


Fig. 6 : Le prototype réalisé avec le CPLD Cypress

On présente, figure 7 et 8, le schéma de la carte utilisant un CPLD Cypress CY37064P44.

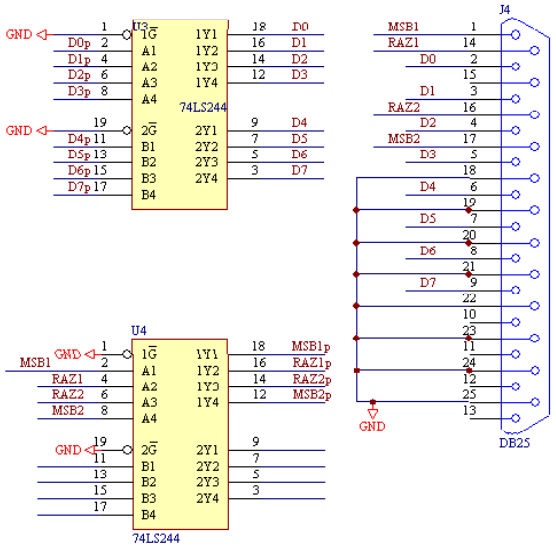


Fig. 7 : La protection des signaux issus du port parallèle du PC

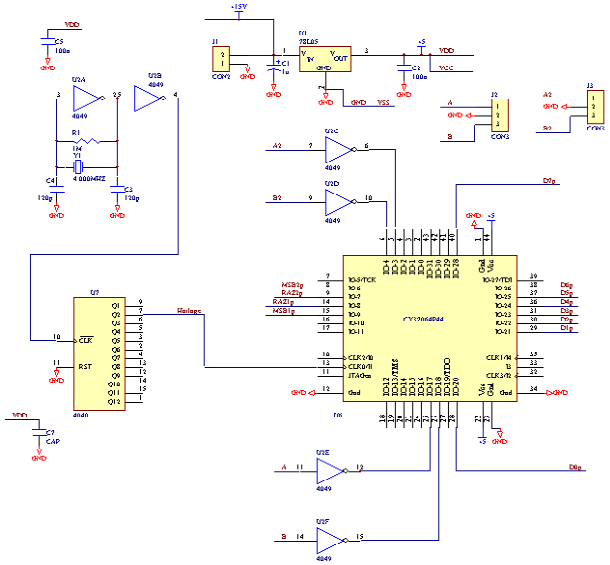


Fig. 8 : Connexion du CPLD aux signaux d'entrée-sortie, obtention de l'horloge

2.7 Remarques pratiques sur la réalisation, la réflexion des signaux rapides

Nous avons aussi réalisé une carte en utilisant un kit de développement proposé par Altera. Dans ce cas le CPLD n'est pas présent sur le prototype mais est relié par une nappe comme le montre la figure 9.

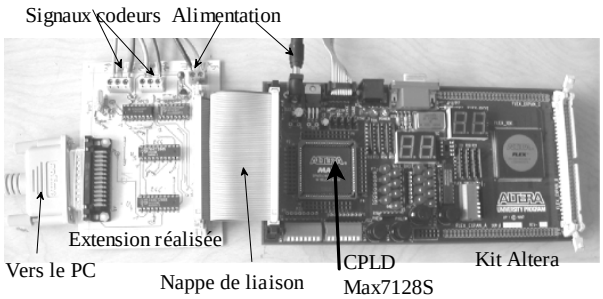


Fig. 9 : La carte d'extension au kit Altera

Les fronts raides des signaux sont alors réfléchis en bout de lignes et on observe l'évolution donnée figure 10 pour le signal MSB (donné à titre d'exemple). Le passage de 0 à 1 donne lieu à une réflexion, le signal logique est perturbé, le fonctionnement de notre application n'est pas fiable alors qu'il est satisfaisant avec le circuit Cypress.

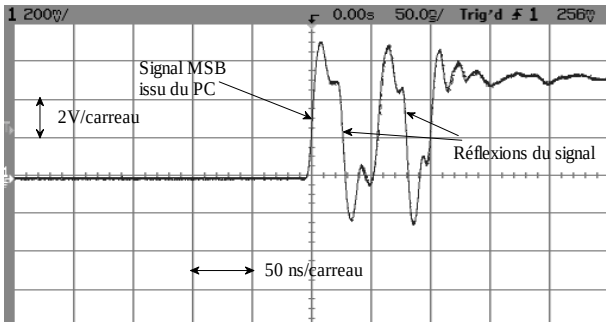


Fig. 10 : Evolution des signaux en cas de réflexion

3 LES ROUTINES EN C D'UTILISATION DE LA CARTE

Il s'agit d'écrire des fonctions qui commandent les 4 signaux RAZ1, RAZ2, N_Codeur, MSB pour déterminer les valeurs de comptage, sur 16 bits, N1 et N2 des 2 codeurs. On rappelle que les 8 bits de poids faibles de N1 sont notés LSB1 et les 8 bits de poids forts sont notés MSB1 (De même N2, MSB2, LSB2 pour le codeur N°2).

La séquence à respecter pour obtenir N1 est la suivante :

1) Définir en entrée le registre d'adresse 0x378 en mettant à 1 le bit 5 du registre 0x37A.

2) Mettre RAZ1 et RAZ2 à 1 pour autoriser le comptage (RAZ1 est le bit 1 du registre d'adresse 0x37A, il est inversé sur le connecteur) (RAZ2 est le bit 2 du registre d'adresse 0x37A).

3) Mettre N_Codeur à 0 (N_Codeur est le bit 0 du registre d'adresse 0x37A, il est inversé sur le connecteur).

4) Mettre MSB à 0 (MSB est le bit 3 du registre d'adresse 0x37A, il est inversé).
5) Lire LSB1 (contenu du registre d'adresse 0x378)
6) Mettre MSB à 1 sans changer les autres bits de 0x37A.
7) Lire MSB1(contenu du registre d'adresse 0x378)
8) Calculer N1 = 256*MSB1 + LSB1.

Pour écrire le mot, Data, défini sur 16 bits, sur un registre d'adresse, Adr, on dispose des routines Data = PortIn(Adr) pour lire et PortOut(Adr, Data) pour écrire.

Les séquences 1 à 4 consistent à écrire,sur le registre 0x37A, le mot de 8 bits défini par bit5 = 1, bit3 = 1, bit2 = 1, bit1 = 0, bit0 = 1 soit le mot binaire XX1X1101 ➔0x2D convient.

La séquence 6 consiste à changer le bit 3, soit à écrire 0x25. D'où la fonction en C :

```
int Lit_Codeur1(void)
{  PortOut(0x37A,0x2D);
   unsigned short int LSB1 = PortIn(0x378) & 255; // masque les 8 bits de poids fort
    PortOut( 0x37A,0x25);
    unsigned short int MSB1 = PortIn(0x378) & 255;
    unsigned short int N1 = (MSB1 << 8) | LSB1;    //idem 256*MSB1 + LSB1
    return N1 ;
}
```

On écrit de la même manière la fonction réalisant la mise à 0 en commandant simplement le bit RAZ1. Et de même pour le codeur N° 2.

4 APPLICATION, IDENTIFICATION D'UN PENDULE

Ici on ne s'intéresse qu'à l'identification du pendule pas à la commande de l'angle lors des déplacements obtenus par le moteur brushless.
L'identification est obtenue par un essai de lâcher. On note α la position angulaire du pendule par rapport à la verticale. Si on écarte le pendule d'angle, inconnu mais assez faible, et que l'on le lâche, à l'instant t_0, inconnu, avec une vitesse initiale inconnue, le retour à l'équilibre est obtenu, (si α reste faible tel que $\sin(\alpha) \approx \alpha$) selon la loi, $\hat{\alpha}(t) = Ae^{-m.\omega_0.(t-t_0)}.\sin[\omega_0\sqrt{1-m^2}.(t-t_0)]$. Le principe de l'identification consiste à enregistrer l'évolution réelle de $\alpha(t)$ et à tracer sur le même graphe la courbe $\hat{\alpha}(t)$ en fixant arbitrairement les paramètres A, t_0, m, ω_0. Par approximations successives on cherche à minimiser l'écart entre $\alpha(t)$ et $\hat{\alpha}(t)$. On pourrait utiliser le critère des moindres carrés et une méthode d'optimisation non linéaire mais l'ajustement manuel de paramètres est formateur pour l'étudiant qui peut juger visuellement de leur influence sur la courbe. Cette démarche est présentée dans la référence [3].

Le logiciel réalisé est écrit en C. La face avant est donnée figure 13. L'utilisateur précise la durée de l'acquisition et la période d'échantillonnage. Puis il lance l'acquisition via le bouton "MESURER" et il écarte le pendule de la position d'équilibre. A la fin de l'enregistrement le bouton "SIMULER" devient actif et l'utilisateur choisit les 4 paramètres A, t_0, m, ω_0. La figure 13 présente un essai d'identification obtenu avec un enregistrement de 5s, les paramètres identifiés apparaissent, on constate que le pendule est très peu amorti, m ≈ 0,004, la pulsation propre est ω_0 ≈ 7,9 rad/s. Ces deux paramètres sont caractéristiques du pendule et ne dépendent pas de l'essai. On peut vérifier pour différentes valeurs de A et t_o que c'est bien le cas.

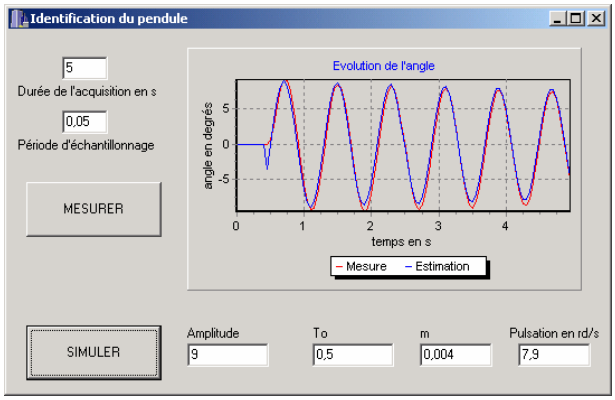
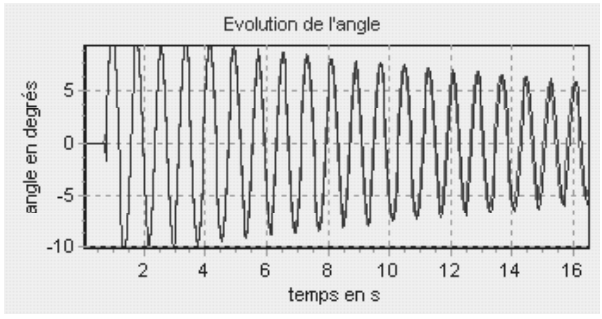
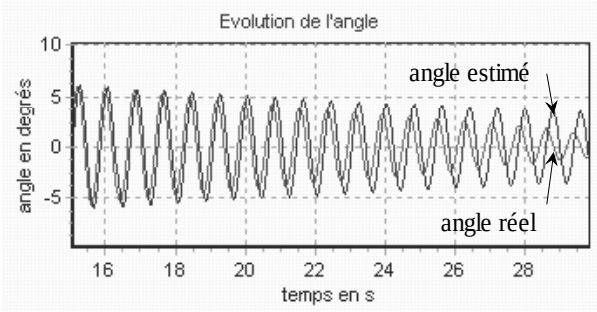


Fig. 13 Un essai d'identification

Sur la figure 14 on présente un essai de validation des paramètres identifiés. On a effectué un essai de durée 30s et on a conservé les valeurs m ≈ 0,004 et ω_0 ≈ 7,9 rad/s déjà identifiées, on a simplement ajustés les paramètres A et t_0. On constate qu'au début de l'essai, pour 0 < t < 15s, l'angle estimé et l'angle réel sont très proches mais à la fin de l'essai 15< t < 30s les deux courbes s'écartent. En effet quand l'amplitude d'oscillation diminue les frottements secs ont une influence croissante et ce phénomène n'est pas pris en compte par la modélisation linéaire du pendule.



L'angle estimé et l'angle mesuré sont presque confondus au début de l'essai



Les frottements secs amènent un écart à la fin de l'essai

Fig. 14 : Comparaison du signal estimé et du signal expérimental

5 CONCLUSION

Ce travail correspond à la formation moderne des techniciens en Génie Electrique. Il permet de fédérer des parties logicielles en VHDL et en C et une partie matérielle. De plus l'application fait appel aux notions d'identification vue en automatique. De nombreuses utilisations de cette interface sont envisageables, comme la commande de position par exemple. La réalisation concrète d'un prototype semble indispensable pour finaliser un travail technique, les seules conceptions logicielles ne permettant pas d'appréhender les difficultés liées à la concrétisation. Cet aspect concret est une motivation pour certains étudiants mais un motif de découragement pour d'autres qui souhaitent surtout s'investir dans des aspects théoriques. La recherche des pannes nécessite souvent beaucoup de temps mais exige d'établir des procédures de tests pour en tirer des diagnostics et cela constitue une compétence obligatoire pour un technicien. La partie concernant l'identification permet d'intéresser aussi les étudiants plus théoriciens que praticiens. Le dispositif expérimental permet de nombreuses autres utilisations, comme la commande du chariot en contrôlant l'angle lors des déplacements ou même la commande du pendule inversé.

Bibliographie

[1] Interface à sortie analogique pour codeur incrémental, pages 36 à 41 D. Jacob, Technologies et formations n° 110 septembre –octobre 2003

[2] Parallel Port Complete, Jan AXELSON, Lakeview research ISBN 096508191-5

[3] Identification à l'aide de la réponse indicielle en TP de régulation. D. Jacob, Génie Electrique Service d'Information, décembre 2002, pages 29 à 35

ANNEXE : Note sur le codeur incrémental

Le codeur incrémental rotatif permet de mesurer une position angulaire. Il est constitué (pour les capteurs optiques) d'un disque comportant *N* secteurs opaques et transparents couvrant l'angle $\Delta\theta = 2\pi/N$. On place d'un coté une source lumineuse et de l'autre deux capteurs photosensibles décalés de l'angle $\Delta\theta/4$ (ou $k.\Delta\theta + \Delta\theta/4$, avec *k* entier). Les détecteurs délivrent ainsi deux signaux binaires (après mise en forme) *A* et *B* qui permettent de déterminer l'angle de rotation disque par rapport à la position initiale. Une piste qui comporte un seul secteur transparent fournit un signal *Z* qui permet d'obtenir une position de référence absolue comme indiqué figure A1.

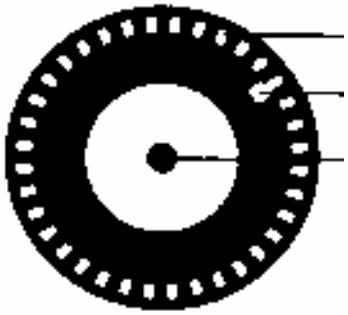


Figure A1 : Disque d'un codeur incrémental rotatif

La position des détecteurs est précisée figure A2. Selon le sens de rotation du disque les signaux *A* et *B* ont des phases différentes.

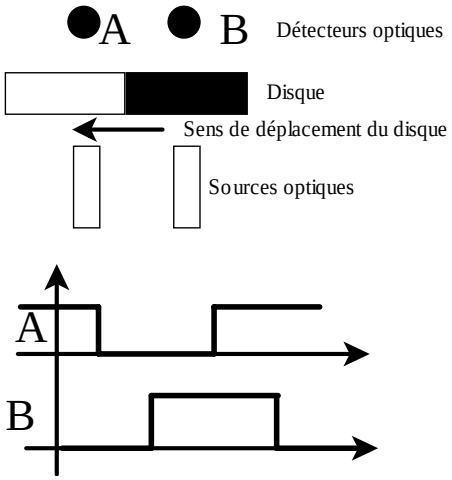


Figure A2 : Principe d'obtention des signaux A et B

Dans la position du disque présentée figure A2 on a A=1 (secteur transparent face au détecteur) et B=0 (secteur opaque face au détecteur). Compte tenu du sens de déplacement du disque le prochain état sera A=0 et B=0 par contre si le disque se déplace dans l'autre sens le prochain état sera A=1 et B=1. Les signaux A et B sont en quadrature, la phase relative des signaux A et B ($+\pi/2$ou $-\pi/2$) permet de déterminer le sens de rotation. En comptant les fronts des signaux on peut obtenir le nombre de pas ($\Delta\theta = 2\pi/N$) dont le disque a tourné.