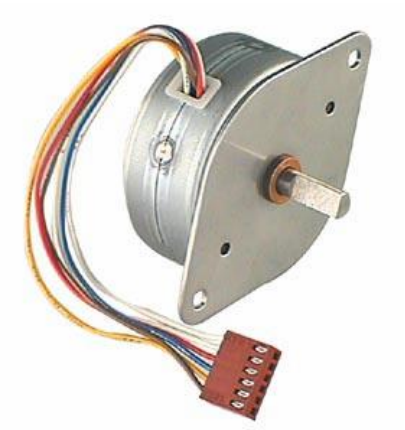
MC60 ï Examen médian
Automne 2009

Etude de la commande dôune Bobineuse

On propose lôétude du logiciel de commande dôune bobineuse électronique. Cette machine est destinée à réaliser des inductances de faibles dimensions et valeurs. Elle est constituée dôun moteur principal dôentraînement du mandrin, en prise directe sans mécanisme de réduction. Le moteur utilisé est un moteur pas à pas courant. Un second moteur, de dimensions plus faibles, est utilisé pour positionner le fil sur le mandrin. La commande de ce second moteur nôest pas prise en compte dans ce sujet. Les caractéristiques principales de ce moteur sont :



- Moteur pas à pas 7,5° (48 pas / tour)
- Utilisable en mode unipolaire et bipolaire sous 12 VDC
- 4 phases 36 W et 2 communs
- Dimensions : Ø 55 x 25 mm, Axe : Ø 6,35 x 17 mm, Entrôaxe de fixation : 67 mm
- 6 fils 48 pas / Tour

Le sujet porte essentiellement sur la commande de ce moteur. Le logiciel embarqué gère la commande des moteurs, la réception des informations de commande à partir de la liaison série RS232 (lien avec un ordinateur de commande), et lôaffichage des informations sur un afficheur LCD de 2 lignes de 16 caractères.

<u>Attention :</u> Les QCM donnent lieu à des points négatifs !!!

Commande électrique du moteur

a) En vous basant sur les caractéristiques fournies, dessinez le schéma électrique des bobines internes du moteur et mentionnez les phases et communs.
b) Chaque phase peut dissiper 36W sous 12V. Quelle est la résistance interne de chaque phase ?
c) On fait fonctionner le moteur en mode unipolaire, sans résistance externe additionnelle. Dessinez le schéma de principe du circuit de commande.
d) On désire alimenter lôensemble sous 24V, afin dôaméliorer les performances du moteur (vitesse maxi). Quelle est la valeur de la résistance à placer entre le commun et lôalimentation ?

Logiciel de pilotage

Le Logiciel étudié est incomplet et figure à la fin du document. Il ne gère pas le moteur de positionnement de lôarrivée du fil à bobiner. Il nôest pas forcément nécessaire de comprendre le rôle exact de chaque instruction.

Description sommaire des modules du logiciel

Le fichier principal comporte les modules suivants :

- RxChar qui attend la réception d'un caractère sur le port série
- GetTemp qui lit une valeur entière utilisée pour la temporisation entre 2 pas moteur
- GetNbtours identique à la précédente, mais qui lit une valeur entière indiquant le nombre de tours à effectuer
- Rotation, hRotation, fonctions qui mettent en mouvement le moteur
- main qui est le programme principal.

Des modules placés dans des fichiers séparés sont utilisés. Pour éviter une perte de temps, ils ne figurent pas dans le sujet. Il est cependant facile de deviner leur utilité et les fonctions contenues. Ces modules sont :

- RS232.c qui contient les fonctions de réglage et de communication série
- delay_f.c qui contient les fonctions d'attente delay
- lcd2l.c qui contient les routines dédiées à l'affichage sur un afficheur LCD deux lignes
- itoa.c qui convertit un nombre entier en chaîne de caractères

Etude du module principal

a) Sur quel port est connecté le moteur ?

b) Quels sont les bits du port moteur disponibles pour une autre fonction (affectation) ?

c) Quel est le rôle de la première ligne du programme ?

d) Que réalise l'instruction `#include "RS232.c"` ?

e) QCM : Quelle affirmation concernant la déclaration `char index;` est exacte ?

1. index est obligatoirement le code ASCII d'un caractère
2. index est un entier dont la valeur est comprise entre 0 et 255
3. index a une valeur comprise entre -128 et +127
4. index est une chaîne de caractères

f) Fonction RxChar : Que réalise l'instruction `while(ndx==pntr);` ?

g) Fonction GetTemp : Quelle opération est en réalité effectuée dans l'instruction suivante : `a=a-'0';` ?

h) Combien de pas fait effectuer au moteur la fonction Rotation lors d'un appel ?

i) Dans la boucle principale de la fonction main, que réalise l'instruction `for(;;)` ?

j) En analysant le module principal, donnez la liste des fonctions susceptibles de figurer dans le module LCD2L.c

k) Compléter la fonction hRotation qui permet de faire tourner le moteur en mode demi-pas.

Module principal bobine1.c

```
// *** Processeur : 16F877 ***
#include "RS232.c"
#include "delay_f.c"
#include "lcd2l.c"
#include "itoa.c"

char index;
char pntr;
int16 nbtours;
char halfstep;

const char step[]={0x20, 0x10, 0x40, 0x80};
const char hstep[]={0x20,0x30,0x10,0x50,0x40,0xC0,0x80,0xA0};

char RxChar(void)
{
char a;
    while(ndx==pntr);
    a=buf[pntr];
    pntr=pntr+1;
    if(pntr==10) pntr = 0;
    Send(a);
    return a;
}

char GetTemp(void)
{
    char a, b;
    b=0;
    a=RxChar();
    while(a>= '0' && a <= '9')
    {
           b = b*10;
           a=a-'0';
           b=b+a;
           a=RxChar();
    }
    if(a != 0x0D)
    {
           b=10;
           Send('$');
           Send(0x0D);
    }
    return b;
}

void GetNbtours(void)
{
char a;

    nbtours=0;
    a=RxChar();
    while(a>= '0' && a <= '9')
    {
           nbtours = nbtours*10;
           a=a-'0';
           nbtours=nbtours+a;
           a=RxChar();
    }
    if(a != 0x0D)
    {
           nbtours=0;
           Send('$');
           Send(0x0D);
    }
}


char tempo;

void Rotation(char Sens)
{
```

```
PORTD=step[index];
delay(tempo);
if(Sens==1)
    index=(index+1)%4;
else
    index=(index-1)%4;
}

void hRotation(char Sens)
{
```

A compléteré

```
}

void main(void)
{
    char a,sens,i;
    int16 k;

    index=0;
    halfstep=0;
    delay(10);
    LcdInit();
    delay(200);
    LcdHome();
    delay(1);
    LcdPrint("BOBINEUSE V2");
    LcdLine2();
    LcdPrint("1200 Bauds,n,8");

    TRISD=0b00001111;

    PORTD=step[index];
    delay(10);
    PORTD=0;

    SerialSetup(12); // 1200 bauds

    tempo=10;
    ndx=0;
    pntr=0;

    RxEnable();
    Send(0x0D); Send(0x0A);
    SerialSendStr("BOBINEUSE V2");
    Send(0x0D); Send(0x0A);

    delay10(100);

    LcdLine2();
    LcdPrint("               ");

    /* Boucle principale */
    for(;;)
    {
           a=RxChar();

           switch (a)
           {
                  case 'f' : // Marche Avant (Faire suivre du Nb de tours +Crlf)
                  sens=1;
                  break;

                  case 'b' : // Marche Arriere
                  sens=0;
                  break;

                  case 's' : // Vitesse (Valeur comprise entre 5 et 255 + Crlf)
                  sens=3;
                  break;

                  case 'r': // Rotation continue
                  sens=1;
```

```
                while(ndx==pntr) hRotation(sens);
                RxChar();
                PORTD=0;
                sens=2;
                break;

                case 'p': // 1/2 pas
                halfstep=1;
                sens=2;
                break;

                case 'P': // Pas complet
                halfstep=0;
                sens=2;
                break;

                default :
                sens=2;
                break;
        }
        if (sens <2) GetNbtours();
        if(nbtours==0) sens=2;

        if (sens == 3) tempo = GetTemp();

        k=0;
        i=0;

        if(halfstep == 0)
                while (k<nbtours && sens < 2)
                {
                        Rotation(sens);
                        itoa(k+1);
                        i++;
                        if (i==48)
                        {
                                i=0;
                                k=k+1;
                                LcdLine2();
                                LcdPrint(ValeurAscii);
                        }
                        if(ndx!=pntr) break;
                }
        else
                while (k<nbtours && sens < 2)
                {
                        hRotation(sens);
                        itoa(k+1);
                        i++;
                        if (i==96)
                        {
                                i=0;
                                k=k+1;
                                LcdLine2();
                                LcdPrint(ValeurAscii);
                        }
                        if(ndx!=pntr) break;
                }

        PORTD=0;
    }
}
```