Contrôle d'Informatique Industrielle

David Delfieu

Polycopié autorisé - 2h00

1 Exposé du problème

On veux réaliser la commande d'une imprimante. Deux moteurs pas à pas sont utilisés pour le déplacement de la tête d'impression. Le moteur M_1 gère les déplacements de type colonne en "x" (horizontal) et le moteur M_2 gère les déplacements de type ligne en "y" (vertical). Les deux moteurs seront commandés en Pas Entier. Deux variables entières x et y sont déclarées de façon globale et modélise la position courante de la tête d'impression.

Une machine à courant continu permet de récupérer le papier du magasin A_4.

2 La tête d'impression

Toutes les fonctions à écrire devront mettre à jour les variables x et y en cohérence avec la position courantre de la tête d'impression.

2.1 Délai (1 pt)

Ecrire une fonction delai permettant de créer des délais multiples d'un dizième de seconde.

```
void delai(int d) {
  int i;
  for (i=0;i<d*100;i++) _delay_ms(i);
}
```

2.2 Déplacement élémentaire du moteur colonne : M_1 (3 pts)

Un accouplement à base de pignon et de crémaillère transforme le déplacement rotatif en déplacement horizontal du moteur M_1. De plus, ce moteur est implanté sur le port D de la façon suivante :

- PD_0 sur la phase 0 du moteur ;
- PD_1 sur la phase 1 du moteur ;
- PD_2 sur la phase 2 du moteur ;

– PD_3 sur la phase 3 du moteur ;

En considérant que le déplacement de 4 pas de M_1 produit un déplacement en "x" d'une unité. Ecrire la fonction *deplace*, dont le prototype est donné par la suite, permettant de réaliser un déplacement de $+/-d$ millimètre(s) dans l'axe horizontal "x" (déplacement en colonne) à partir de la position courante "x". Entre chaque pas on observera un délai de 3 dixièmes de seconde.

```
int x,y;
void deplace(int d) {
  int i,j;
  int PE[4]={1,2,4,8};

  if (d>0){
    for(j=1;j<d;j++)
      for (i=0;i<4;i++) {
        PORTD=PE[i];
        delai(3);
      }
    x=x+d;
  }
  else {
    d=-d;
    for(j=0;j<d;j++)
      for (i=3;i>=0;i--) {
        PORTD=PE[i];
        delai(3);
      }
    x=x-d;
  }
}
```

2.3 Remise en position 0 de M_1 (1 pt)

Ecrire la fonction *RemiseZero* dont le prototype est donné par la suite, permettant de remettre M_1 en position zéro c'est à dire correspondant à la première colonne.

```
void RemiseZero(int d) {
  int zero=-x;
  deplace(zero);
  x=0;
}
```

2.4 Nettoyage de la tête d'impression par M_1 (2 pts)

On suppose que le nombre de colonnes d'impression est défini par l'étiquette "N". Ecrire la fonction *Nettoyage* dont le prototype est donné par la suite, permettant les actions suivantes décrites dans l'algorithme suivant :

```
Retour en position zéro;
pour les N colonnes faire
    Un déplacement de 1 millimètre ;
    Faire une temporisation de 10 millisecondes ;
fin
Retour en position zéro;
```

```
void Nettoyage(void) {
  int i;
  RemiseZero();
  for(i=0;i<N;i++) {
    deplace(1);
    delai(1);
  }
  RemiseZero();
}
```

2.5 Déplacement du moteur ligne : M_2 (4 pts)

Un accouplement à base de pignon et de crémaillère transforme le déplacement rotatif en déplacement vertical du moteur M_2. De plus, ce moteur est implanté sur le port D de la façon suivante :

- PC_4 sur la phase 0 du moteur ;
- PC_5 sur la phase 1 du moteur ;
- PC_6 sur la phase 2 du moteur ;
- PC_7 sur la phase 3 du moteur ;

En considérant que le déplacement d'une ligne du moteur M_2 correspond à 8 pas écrire la fonction *Skip* permettant de faire faire au moteur M_2 le saut de p lignesEn considérant que le déplacement d'une ligne du moteur M_2 correspond à 8 pas écrire la fonction *Skip* permettant de faire faire au moteur M_2 le saut de p lignes

```
void Skip(int p) {
  int i,j;
  int PE[4]={16,32,64,128};
  if (p>0) {
    for(j=0;j<2*p;j++)
```

```
        for (i=0;i<4;i++)
          PORTD=PE[i];
      y=y+p;}
      else {p=-p;
        for(j=0;j<2*p;j++)
          for (i=3;i>=0;i--)
            PORTD=PE[i];
        y=y-p;}
      }
  }
```

2.6 L'entraînement du papier

Le moteur d'avance du papier est commandé par une Modulation de Largeur d'Impulsion (MLI ou PWM) via un hacheur. Le hacheur est lié à la patte $0C1_B$ soit la patte PB_2 de l'ATMEGA. Ecrire un programme principal dont un algorithme est donné ci-dessous ainsi que la fonction engagePapier() qui permettront de réaliser une PWM à phase et fréquence correcte fixe de 60% commandant de fait une vitesse constante pour la prise de feuille dans le magasin de feuilles A_4 jusqu'a ce qu'un capteur indique qu'une feuille est engagé devant la tête d'impression et est prête. Ce Capteur est connecté à la patte PD_2 correspondant à l'interruption externe INT_0.

Initialisations;
tant que *Boucle inifinie* **faire**
  attenteOrdreImpression();
  engagePapier();
fin

Algorithme 1 : Programme principal

attenteOrdreImpression() {
Fonction système déjà définie;
}

Fonction `Système`

```
  #define F_CPU 12E6   //quartz carte
  #include <avr/interrupt.h>
  #include <util/delay.h>   //delay_ms
  #define Thash 1000
  #define VRAI 0
  #define FAUX 1
```

engagePapier() {
Initialisations de la MLI à phase et à fréquence correcte;
tant que *Variable globale indiquant "feuille prête" à faux* **faire**
| rien();
fin
arrêter la MLI;
}

Fonction à définir

$INT_0\_vect()$ { ...}

Fonction à définir

```
int x,y;
int arretEngage;

// IT externe 1 pt
ISR(INT0_vect){
  arretEngage=VRAI;
}

// engage 4 pt
void engagePapier(void) {
   TCCR1A=0x21;  // 0 b 0010 0001
   TCCR1B=0xB1;  // 0 b 1101 0100
   OCR1A=Thash;  // valeur max
   OCR1B=600;    // rapport cyclique
   while(arretEngage==FAUX) ;
   TCCR1B=0;
}

int main (void)
{
  // Ports 1 pt
   DDRB = 0xFF;  // port B en sortie
   DDRD = 0xFF;  // PD0 en entree
   arretEngage=FAUX;

   // interuptions 2 pt
   GICR  |=  (1<<INT0); // en particulier INT0 validée
   MCUCR |=  (1<<ISC01) | (1<<ISC00); //ISC11 ISC10 ISC01 ISC00
```

```
            = 0011 front montant
        sei();  // authorisation des interruptions

        // Boucle 1
        while(1){
          attenteOrdreImpression();
          engagePapier();
        }
        return(0);
    }
```