Contrôle de Micro 8 bits

2 heures - David Delfieu

Vendredi 29 mai 2009

Introduction

On veut commander le moteur pas à pas implanté sur la maquette vue en TP mais suivant un modèle de grafcet. Comme dans le TP 2, le moteur sera cablé de la façon suivante :

- PB_0 sur la phase 0 du moteur ;
- PB_1 sur la phase 1 du moteur ;
- PB_2 sur la phase 2 du moteur ;
- PB_3 sur la phase 3 du moteur ;

Concernant la partie commande :

- PD_0 sur un bouton poussoir appelé Dcy ;
- PD_1 sur un bouton poussoir appelé PE ;
- PD_2 sur un bouton poussoir appelé DP ;
- PD_3 sur un bouton poussoir appelé F ;

De plus, on connecte 4 Leds aux sorties du port C :

- LAR sur PC_0 ;
- LA sur PC_1 ;
- LPE sur PC_2 ;
- LDP sur PC_3 ;

1 Exercices

1.1 Delai

Ecrire une fonction delai permettant de générer des temporisations variables multiples de 20 ms.

Solution

```
void delai(int d){
  int i;
  for(i=0;i<d;i++) _delay_ms(20);
}
```

1.2 Pas Entier

On va définir dans cet exercice une fonction *entier* permettant de faire tourner le moteur pas à pas en pas entier dont le sens est défini par la variable *sens*. Dans cette fonction on va tenir compte de l'état

d'une variable globale [1] que l'on nommera *tourne*. La fonction fait tourner le moteur si cette variable vaut 1. Par contre dès que cette variable re-passe à zéro, le moteur s'arrête et l'on sort de la fonction.

```
void entier(int sens);
```

Solution

```
void entier(int sens){
  int i,j;
  int Tentier[4]={1,2,4,8};
  while(tourne==1) {
  if (sens==HORAIRE)
    for(i=0;i<4;i++){
      delai(10);
      PORTB=Tentier[i]; // 1 2 4 8
    }
  else
    for(i=3;i>=0;i--){
      delai(10);
      PORTB=Tentier[i]; // 1 2 4 8
    }
  }
}
```

1.3 Demi Pas

On va définir dans cet exercice une fonction *demipas* permettant de faire tourner le moteur pas à pas en pas en demi-pas dont le sens est défini par la variable *sens*. Dans cette fonction on tiendra compte de l'état de la variable globale *tourne*. La fonction fait tourner le moteur si cette variable vaut 1. Par contre dès que cette variable re-passe à zéro, le moteur s'arrête et l'on sort de la fonction.

```
void demipas(int sens);
```

Solution

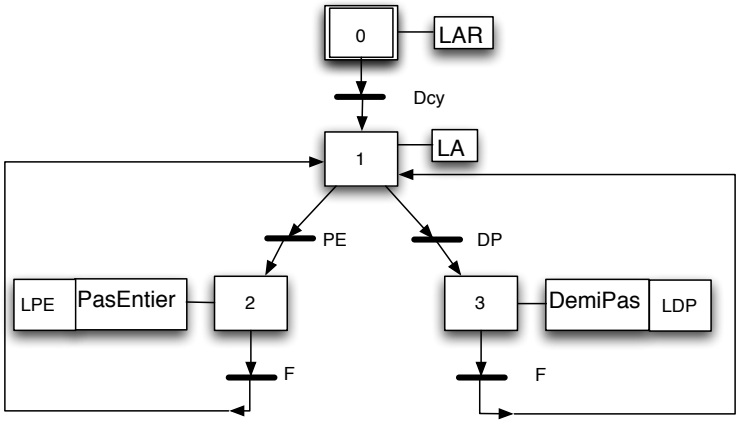
```
void demipas(int sens){
  int i,j;
  int Tdemi[8]={1,3,2,6,4,12,8,9};
  while(tourne==1) {
  if (sens==HORAIRE)
    for(i=0;i<8;i++){
      delai(10);
      PORTB=Tdemi[i]; // 1 2 4 8
    }
  else
    for(i=7;i>=0;i--){
```

1. déclarée en dehors du main et des fonctions

```
            delai(10);
            PORTB=Tdemi[i]; // 1 2 4 8
          }
        }
      }
```

2 problème

Le grafcet de commande est représenté dans la figure (fig. 2) suivante :



Ce grafcet comprend 4 entrées capteurs reliées aux bouton poussoirs (de type monostable) Dcy, PE, DP et F.

- Dcy : Départ cycle,
- PE : commande du moteur en pas entier ;
- DP : commande du moteur en demi-pas ;
- F : arrêt du moteur.

La partie actionneur constitue la commande du moteur pas à pas en mode "pas entier" ou "demi-pas". De plus, quatre leds témoignent de l'activité du processus LAR, LA, LPE et LDP.

- LAR : Led indiquant que le système est arrété ;
- LA : Led d'Attente ;
- LPE : Led témoignant que le moteur tourne en Pas Entier ;
- LDP : Led témoignant que le moteur tourne en Demi-Pas ;

2.1 Initialisations

En considérant les initialisations suivantes (similaires au TP), retrouver les initialisations des différents PORTS correspondant à ce problème.

```
#define Dcy   1 // entrées
#define PE    2
#define DP    4
#define LAR   0 //sorties
#define LA    1
#define LPE   2
#define LDP   3
```

Dans cette application particulière le cas du capteur F sera traité de façon différente : F est cablée sur l'entrée d'interruption externe INT_1 : c'est à dire PD_3. Ce signal sera considéré actif sur un front

montant. Donc on va utiliser cette interruption externe combinée à la variable *tourne* pour réaliser la sortie des étapes 2 et 3.

Solution

```
int t;
DDRD=0x00;   // 1111 0111 PD2 : interruption Opto
DDRB=0x0F;
DDRC=0x0F;
PORTB = 0;
GICR  |= (1<<INT1); // en particulier INT0, validée
MCUCR |=   (1<<ISC11) | (1<<ISC10); // front montant
sei(); // Enable global interrupts
```

2.2 Réflexion préliminaire

Pour le traitement en OU de la sortie de l'étape 1, le plus aisé est d'écrire une nouvelle fonction transition dont voici le prototype :

```
int transition_OU(int t1, int t2) ;
```

1. Donnez le code de la nouvelle fonction *transition_OU* de la sortie de l'étape 1.
2. Réfléchissez et indiquer de façon littérale comment vous aller procéder pour gérer la sortie des étapes 2 et 3.

Solution

```
int transition_OU(int t1, int t2) {
  int c1=0,c2=0;
  do {
    c1 = PINC;c2=c1;
    c1 = capteur & t1;
    c2 = capteur & t2;
  } while((c1 != t1) && (c2 != t2));
  if (c1==t1) return 1;
  else return 2;
}
```

La sortie des étapes 2 et 3 est commandée par la variable tourne qui est mise à zéro par le sous-programme d'interruption associé à l'interruption externe liée au bouton F. Cette variable sera mise à un dans l'étape 1.

2.3 Le programme

En considérant les instructions suivantes, écrire le programme qui réalise le grafcet. NB : En pas entier le moteur tournera dans le sens HORAIRE et en demi-pas le moteur tournera en sens TRIGO.

```
void transition(int condition) { ...}
int transition_OU(int t1, int t2) {...}
void etape0(void) {  ...}
int main(void)
```

```
{
  ...  // initialisations en E/S des PORTS
  etape0();
  transition(...);
  ...
}
```

Solution

```
int main(void){
  int t;
  DDRD=0x00;   // 1111 0111 PD2 : interruption Opto
  DDRB=0x0F;
  DDRC=0x0F;
  PORTB = 0;
  GICR  |= (1<<INT1); // en particulier INT0, validée
  MCUCR |=   (1<<ISC11) | (1<<ISC10); //  front montant
  sei(); // Enable global interrupts
  do {
    etape0();
    transition(Dcy);
    etape1();
    t=transition_OU(PE,DP);
    if (t==1) etape2();
    else etape3();
  } while(1);
  return(0);
}
```

Documentation technique

Le vecteur d'interruption se nomme $INT1\_vect$. La primitive de placement des ITs est $ISR(...)\{...\}$. Le type d'événement susceptible de déclencher une interruption externe se programme à l'aide des bits $ISC_{11}, ISC_{10}, \in MCUCR$

$MCUCR$ contient les bits de gauche à droite suivant :

$$MCUCR = SE\ SM_2\ SM_1\ SM_0\ ISC_{11}\ ISC_{10}\ ISC_{01}\ ISC_{00}$$

Pour les bits ISC_{11}, ISC_{10}, on a :

0 0 : The low level of INT1 generates an interrupt request.

0 1 : Any logical change on INT1 generates an interrupt request.

1 0 : The falling edge of INT1 generates an interrupt request.

1 1 : The rising edge of INT1 generates an interrupt request.

L'interruption externe INT_1 se valide à l'aide du bit $INT_1 \in GICR$:

$$GICR = INT_1\ INT_0 - - - - IVSEL\ IVCE$$

Les interruptions se valident de façon générale par le bit $I \in SREG$.

$$SREG = I\ T\ H\ S\ V N\ Z\ C$$