Travaux Pratiques Micro 8 bits
La station de fabrication de béton

David Delfieu - Vincent Pauvert

14 mars 2008

Introduction

Une maquette modélise la fabrication de béton, par un dosage de sable et de ciment. Ce processus est modélisé par le $GRAFCET$ suivant :

Une maquette comporte 6 sorties : $PA_0, PA_1, PA_2, PA_3, PA_4, ALARM$ correspondant aux interrupteurs de la carte. Ces derniers premettent de valider les différentes tranisitions du GRAFCET. Les entrées $PB_0, PB_1, PB_2, PB_3, PB_4, PB_5, PB_6, PB_7$ servent à commander les leds simulant le processus de fabrication de béton.

1 Initialisations

1. Créer un projet ;
2. Dans votre fichier c principal : tp1.c, éditer les étiquettes suivantes :

```
#include <avr/io.h>
#define F_CPU 12E6  // quartz carte
#include <avr/interrupt.h>
#include <util/delay.h> // delay_ms

#define DEPART_CYCLE  1    // entrées
#define TREMIE_VIDE   2
#define MALAXEUR_VIDE 4
#define POIDS_ATTEINT   8
#define CAMION_DISPO    16
#define ALARME      32

#define POSITION_ATTENTE    0 // sorties
#define VANNE_C       0
#define VANNE_S       1
#define MALAXEUR_M      3
#define VANNE_W         4
#define VANNE_M       5
#define PROCESSUS_COURS 6
#define STATION_ARRETEE   7
```

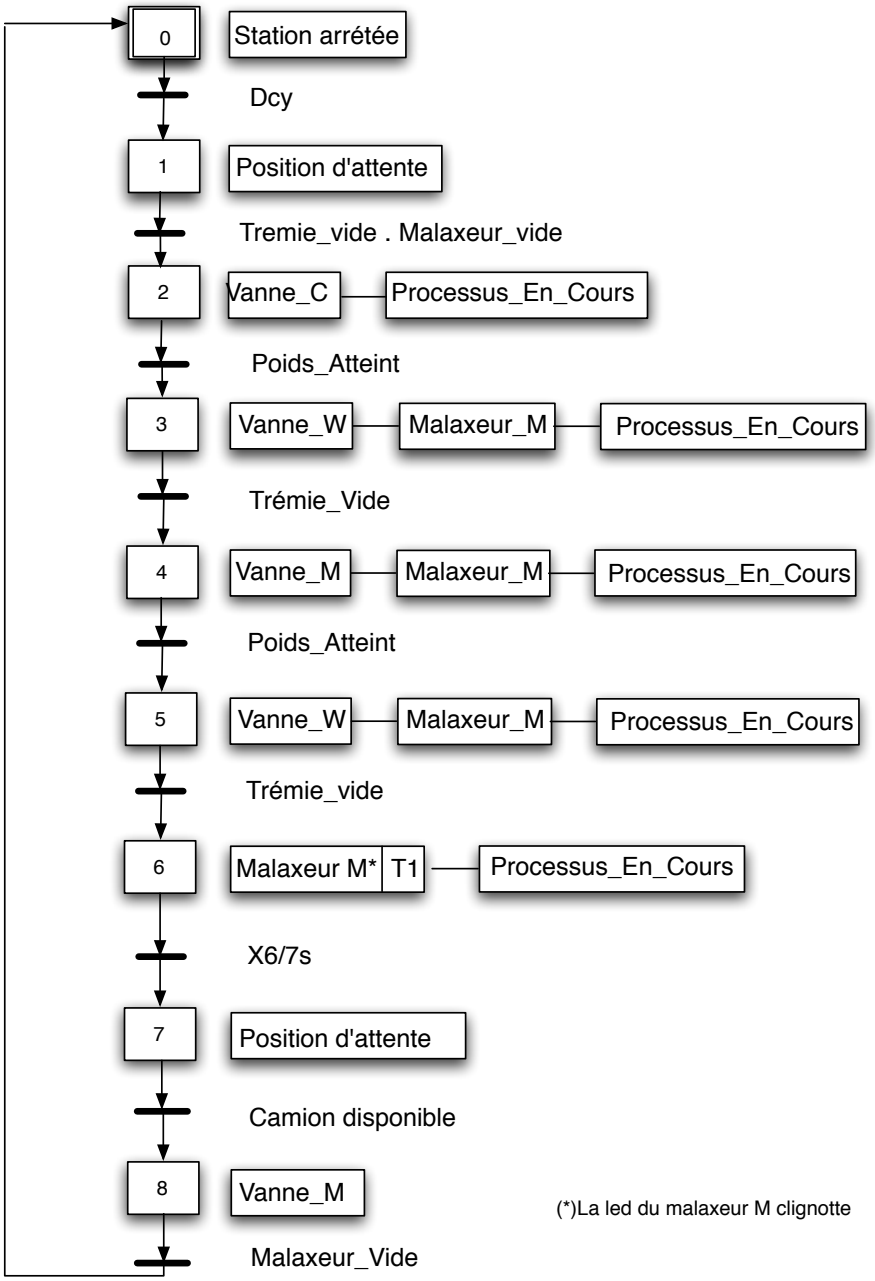

0
Station arrétée
Dcy
1
Position d'attente
Tremie_vide . Malaxeur_vide
2
Vanne_C
Processus_En_Cours
Poids_Atteint
3
Vanne_W
Malaxeur_M
Processus_En_Cours
Trémie_Vide
4
Vanne_M
Malaxeur_M
Processus_En_Cours
Poids_Atteint
5
Vanne_W
Malaxeur_M
Processus_En_Cours
Trémie_vide
6
Malaxeur M*
T1
Processus_En_Cours
X6/7s
7
Position d'attente
Camion disponible
8
Vanne_M
(*)La led du malaxeur M clignotte
Malaxeur_Vide

FIG. 1 – G7

3. Effectuez les cablages des sorties de la maquette sur le port C et PD_2 du μC :
   - PA_0 : départ cycle sur PC_0 ;
   - PA_1 : trémie vide sur PC_1 ;
   - PA_2 : malaxeur vide sur PC_2 ;
   - PA_3 : poids atteint sur PC_3 ;
   - PA_4 : camion disponible sur PC_4 ;
   - $ALARM$ sur PD_2.
4. Effectuez les cablages des entrées de la maquette sur le port D et PB_0 du μC :
   - PB_0 position d'attente sur PB_0 ;
   - PB_1 vanne C sur PD_0 ;
   - PB_2 vanne S sur PD_1 ;
   - PB_3 malaxeur M sur PD_3 ;
   - PB_4 vanne W sur PD_4 ;
   - PB_5 vanne M sur PD_5 ;
   - PB_6 processus en cours sur PD_6 ;
   - PB_7 station arrêtée sur PD_7.
5. Relier la masse de la carte à celle de l'alimentation.
6. Avec les etiquettes précédement saisies, éditez, compiler et tester le programme suivant :

```
void transition(int condition) {
  int capteur=0;
  do {
    capteur = PINC;
    capteur = capteur & condition;
  } while(capteur != condition);
}

void etape0(void){
  PORTD=(1<<STATION_ARRETEE);
}

int main(void)
{
   DDRC = 0x00;   // port C en entrée
   DDRB = 0x01;   // port B : PB0 en sortie
   DDRD = 0b11111011;   // port D en sortie
       sauf  PD2
             // PD2 : entrée d'interruption
                externe
  etape0();
  transition(DEPART_CYCLE);
  ...
}
```

2 Le grafcet

A partir de l'exemple ci-dessus, compléter le programme principal pour mettre en oeuvre le $GRAFCET$ (voir la figure 1).

3 Temporisation

Le clignotement de la led $MALAXEUR_M$ de l'étape 6 doit être réalisé à avec une fréquence précise de $1hz$.

4 Arrêt d'urgence

Modifier le programme de façon à pouvoir traiter un arrêt d'urgence. Le vecteur d'interruption se nomme $INT0\_vect$.

Le type d'événement susceptible de déclencher une interruption externe se programme à l'aide des bit $ISC_{01}, ISC_{00}, \in MCUCR$. Tandis que cette interruption se valide à l'aide du bit $INT_0 \in GICR$.

Le sous-programme associé à cette interruption devra faire clignoter l'ensemble des leds commandables pendant 5 secondes, puis reprendre le $GRAFCET$ là, où il a été interrompu.

5 Processus En cours

Nous allons modifier la prise en compte de la led $PROCESSUS\_EN\_COURS$. Celle-ci va dorénavant, clignoter tout au long du déroulement des étapes $X2$, $X3$,$X4$, $X5$,$X6$. Vous mettrez pour cela en place une interruption de débordement du timer 0.

Rappels :

- Masque d'IT : $TOIE_0 \in TIMSK$,
- Valeur initiale de comptage $TCNT_0$
- $TCCR_0$ contrôle le prédiviseur de fréquence.

Travaux Pratiques Micro 8 bits
Le moteur pas à pas

David Delfieu - Vincent Pauvert

26 mars 2008

Introduction

Ce moteur comporte 48 pas correspondant à des secteurs angulaires identifiés sur une corolle. Le secteur 1 contient un ajourement rendant ainsi détectable la position zéro du moteur. La maquette supportant le moteur pas à pas comprend :

- 4 bits de commande ;
- un connecteur d'alimentation et de masse ;
- une borne $COMPT$ qui passe à un dès que l'optocoupleur détecte la position 1 du moteur pas à pas. Cet état est mémorisé grâce à une bascule D $(74HCT74)$.
- Une borne de Reset permettant de remettre à zéro la bascule D ;
- un interrupteur marche/arrêt permettant de couper l'alimentation du moteur.
- un pottard permet de délivre une tension variable $V \in [0, 5]$ entre la borne se trouvant près de l'optocoupleur et la masse.

Dans ce sujet on abordera la commande du moteur pas à pas.

1 Initialisations

1. Créer un projet ;
2. Dans votre fichier c principal : tp2.c, éditer les étiquettes suivantes :

```
#include <avr/io.h>
#define F_CPU 12E6  // quartz carte
#include <avr/interrupt.h>
#include <util/delay.h> // delay_ms
```

3. Effectuer les cablages suivant :
   - PB_0 sur la borne 0 ;
   - PB_1 sur la borne 1 ;
   - PB_2 sur la borne 2 ;
   - PB_3 sur la borne 3 ;
   - PD_2 sur la borne $COMPT$;
   - PB_4 sur la borne RST ;
4. Relier la masse de la carte à celle de l'alimentation.

2 Commande moteur

2.1 Fonction delai

Ecrire une fonction permettant de faire des delais variables par mutliples de 1 ms.

2.2 Pas entier

Utiliser la fonction delai précédente pour réaliser une fonction permettant de faire tourner le moteur en pas entier à vitesse constante dans le sens horaire de 48 pas.

2.3 demipas

Utiliser la fonction delai précédente pour réaliser une fonction permettant de faire tourner le moteur en demi-pas dans le sens trigonometrique de 48 pas.

2.4 changement de mode

A l'aide d'un bit défini en entrée d'un port, écrire une fonction permettant de changer de mode : pas entier sens horaire vers le mode demi-pas sens trigonométrique (et vice-versa).

3 Convertisseur Analogique Numérique

Relier la borne du pottard à PC_0.

- Réaliser les initialisations dans le $main$ pour mettre en oeuvre une interruption de conversion analogique.
- Utiliser la valeur de conversion pour déterminer le delai utilisé pour régler la vitesse du moteur.
- Avec l'oscilloscope, comparer les vitesses maximales atteintes par les modes demi-pas et pas entier.

4 Optocoupleur

4.1 Mise en oeuvre

L'optocoupleur est géré par 2 bornes : $Compt$ et RST. $Compt$ mémorise (bascule D : $74HCT74$) le fait que le "trou" soit passé devant l'opto. Cependant pour que cette info reste pertinante, il faut remettre ce bit à zéro après la lecture de $COMPT$ par un $pulse$ sur RST. Un $pulse$ est un niveau haut de 20 ms suivi d'un niveau bas.

- Réaliser les initialisations dans le $main$ pour mettre en oeuvre une interruption externe INT_0.
- Dans le sous-programme d'IT commutez le bit PD_0.

Rappels : Registres du Convertisseur

- $ADMUX$: contrôle de la voie, ...
- $ADCSRA$: démarrage, IT, ...
- $ADCH$, $ADCL$: résultat (lire l'octet de poids faible en premier)

Vecteur d'IT du convertisseur : ADC_vect

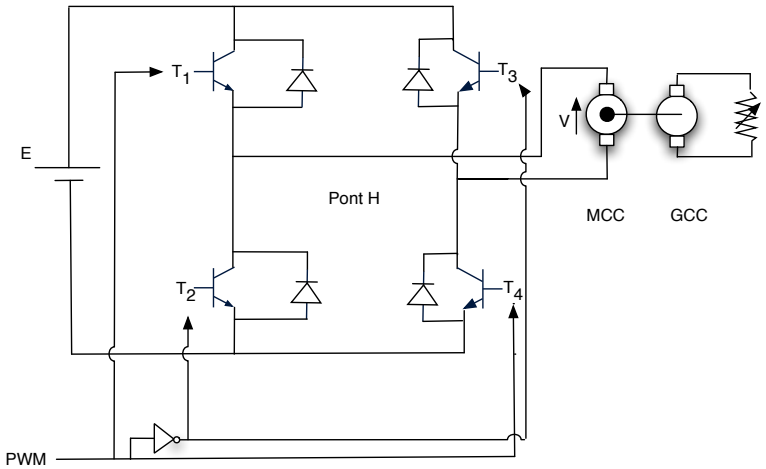
Travaux Pratiques Micro 8 bits
La PWM

David Delfieu - Vincent Pauvert

7 avril 2008

Introduction

On veux commander une Machine à Courant Continu (MCC) par le biais d'un hacheur. Une génératrice est couplée au moteur, elle se comporte comme un frein grâce à un ensemble de résistances variables. Le schéma de principe est donné dans la figure suivante :



On 4 transistors qui sont appairés en 2 couples (T_1, T_4) et (T_2, T_3). Ces deux couples sont reliés à une même ligne de commande PWM, mais le second couple à travers une porte inverseuse. Donc suivant le niveau logique sur cette ligne on a soit (T_1, T_4) passant soit (T_2, T_3) passant. Si (T_1, T_4) sont passant alors (T_2, T_3) sont bloqués et on a une tension de $+E$ aux bornes du moteur, sinon on a $-E$.

L'objet de ce TP est de réaliser une commande en boucle ouverte de la MCC. Soit T une période appelée période de hachage, un rapport cyclique : rapport $\frac{T_{ON}}{T}$ permet la commande de la MCC.
La PWM permet à l'aide d'un signal $v \in [0, 5]$ de réaliser n'importe quel tension moyenne entre $[-E, +E]$, comme le montre la figure suivante :
Ce qui donne pour une période T :

$$V_{moy} = \frac{1}{T}(\int_0^{T_{on}} E.dt + \int_{T_{on}}^{T} -E.dt)$$

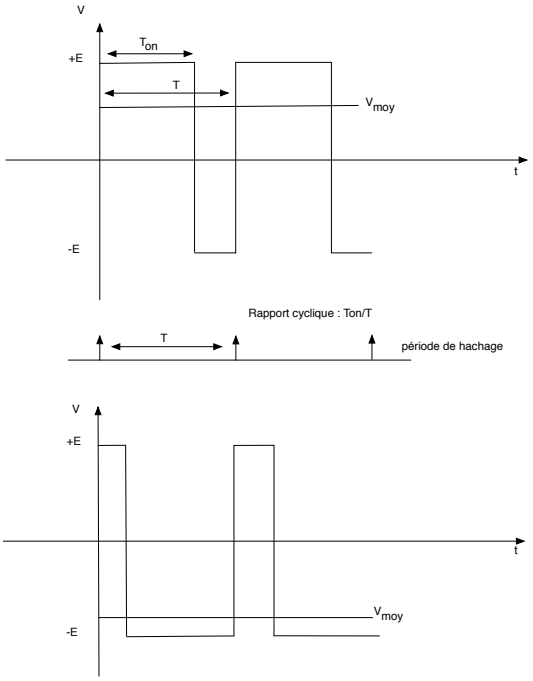


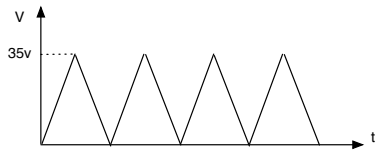
FIG. 1 – Obtention d'une tension moyenne à partir d'un rapport cyclique

d'où

$$V_{moy} = \frac{E}{T}(2.T_{on} - T)$$

1 Commande triangulaire

On veux dans un premier temps commander la MCC avec une commande de type triangulaire du type : On veut de façon périodique provoquer une accélération suivie



d'une décélération. Pour cela vous utiliserez le $TIMER_1$ en mode "phase et fréquence correcte", on utilisera la patte $0C1B$ pour générer la PWM.

2 Commande analogique

Modifier le programme précédent pour commander la MCC à l'aide du potard de la carte μC et à l'aide du module de conversion analogique.