

<b>Fiche élève :</b>  <b>Servomoteur</b>	<b>TS SI</b>		<b>P.P.E Mini serre</b>	
		<h2 style="margin: 0;">Commande d'un servomoteur</h2>		

Nom :	Classe :	Groupe :
-------	----------	----------

**Objectif :** Commander la position de l'axe d'un servomoteur.

**Matériels**  
 Carte ATMELSSI V1 + Module interface ATINY13 pour SSI + Carte ATINY13 + Alimentation 10V.  
 Servomoteur

**Logiciels**  
 CodeVisionAvr.

Sur le site WebGE à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)  
 Schémas de la carte ATMELSSI V1.  
 Infos sur les servomoteurs : [http://fribotte.free.fr/bdtech/pic/pic\\_et\\_servo.html](http://fribotte.free.fr/bdtech/pic/pic_et_servo.html)

### A) Présentation

#### A1) Schéma fonctionnel

Le déplacement des trappes de la serre peut être réalisé avec un servomoteur. Le présent document doit vous permettre de commander un tel dispositif avec un µC.

**Chaîne d'information**

```

graph LR
    E[E : Signal TOR de commande du déplacement.] --> ACQUERIR[ACQUERIR]
    ACQUERIR --> TRAITER[TRAITER]
    TRAITER --> COMMUNIQUER[COMMUNIQUER]
    COMMUNIQUER --> OC0A[OC0A : signal M.L.I (PWM) commandant la position de l'axe du servomoteur]
          
```

**Chaîne d'énergie**

```

graph LR
    DDP[DDP continue] --> ALIMENTER[ALIMENTER]
    ALIMENTER --> DISTRIBUER[DISTRIBUER]
    DISTRIBUER --> CONVERTIR[CONVERTIR]
    CONVERTIR --> TRANSMETTRE[TRANSMETTRE]
    TRANSMETTRE --> DEPLACER[Déplacer les trappes]
          
```

Trappes fermées (θ=θmin)

↓

**Déplacer les trappes**

↓

Trappes ouvertes (θ=θmax)

Fiche servo serre

### Commande de l'axe d'un servomoteur

Révision le mardi 4 mars 2008

La fonction « Traiter » est assurée par un programme implanté dans le microcontrôleur ATMEL ATINY13.

Le signal MLI(PWM) est issu d'une structure appelée « Timer ». Celle-ci est intégrée au microcontrôleur ATMEL.

L'ensemble des structures matérielles étant réunies sur la carte « ATMEL », votre travail va se limiter à la **réalisation du logiciel** à implanter dans le microcontrôleur ATINY13.

Pour cela, vous allez **créer et configurer un projet** avec le magicien du cross-compileur **CodeVisionAVR**. Puis, vous complèterez la structure de ce projet avec les fonctions nécessaires à la mise en œuvre du servomoteur.

**La suite de ce document décrit le travail à réaliser étape par étape. A la fin de cette activité, vous serez capable régler la position de l'axe d'un servomoteur.**

Il faut connaître les caractéristiques du signal à envoyer au servomoteur pour programmer l'ATINY13. Ces informations sont disponibles sur de nombreux sites Internet tels que celui cité en référence.

## **A2) Caractéristique du signal de commande d'un servomoteur**

Dessinez ci-dessous le signal de commande d'un servomoteur, précisez ses caractéristiques (fréquence, tension, rapport cyclique... )

Sachant que l'angle  $\theta$  de l'axe du servomoteur est proportionnel au rapport cyclique  $\alpha$  du signal de commande, représentez et déterminez l'expression de  $\theta = f(\alpha)$  pour une période du signal de commande  $T = 13.6\text{ms}$



### A3) Génération d'un signal Modulé en Largeur d'Impulsion (M.L.I)

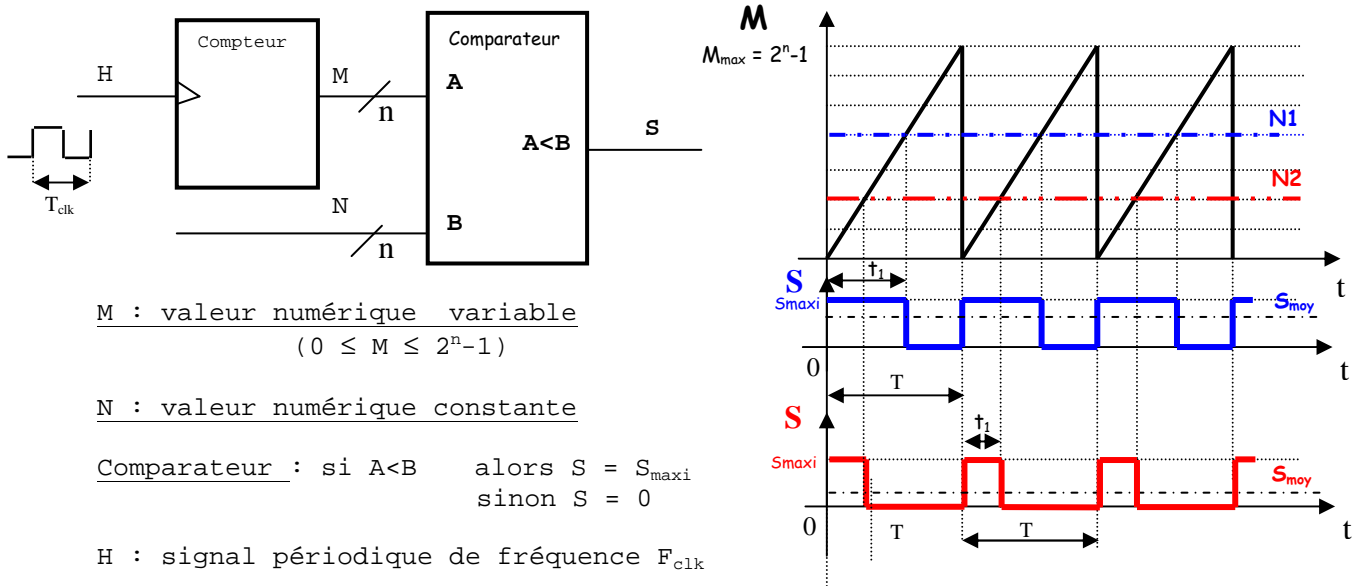
#### • Principe

Le signal de commande d'un servomoteur est dit « **MLI (ou PWM)** ».

(P.W.M.: Pulse With Modulation)

Un signal modulé en largeur d'impulsion peut être obtenu à partir d'un signal périodique  $H$  de fréquence fixe  $F_{clk} = 1/T_{clk}$ . En effet, en appliquant ce signal à l'entrée d'un compteur, on obtient un signal numérique  $M$  (codés sur  $n$  bits) capable d'évoluer entre 0 et  $2^n - 1$ . La représentation de  $M(t)$  est appelée **rampe numérique**. En appliquant  $M(t)$  et un signal constant  $N(t)$  (codé sur  $n$  bits) à un comparateur numérique, on obtient un signal binaire  $S(t)$  de période  $T = (2^n - 1) \cdot T_{clk}$  dont le temps  $t_1$  (à l'état « 1 ») est réglé avec la valeur de  $N$ . On appelle  $\alpha = t_1/T$  le rapport cyclique du signal  $S(t)$ . On montre que la valeur moyenne  $S_{moy}$  de  $S(t)$  est égale au produit de  $\alpha$  par  $S_{maxi}$ .

On donne ci-dessous le schéma de principe d'une structure générant un signal M.L.I et les chronogrammes de  $S(t)$  pour deux valeurs particulières de  $N$ .



Dans les microcontrôleurs, les signaux modulés en largeur d'impulsion sont générés par une structure appelée **TIMER**. Celle-ci répond au principe développé ci-dessus.

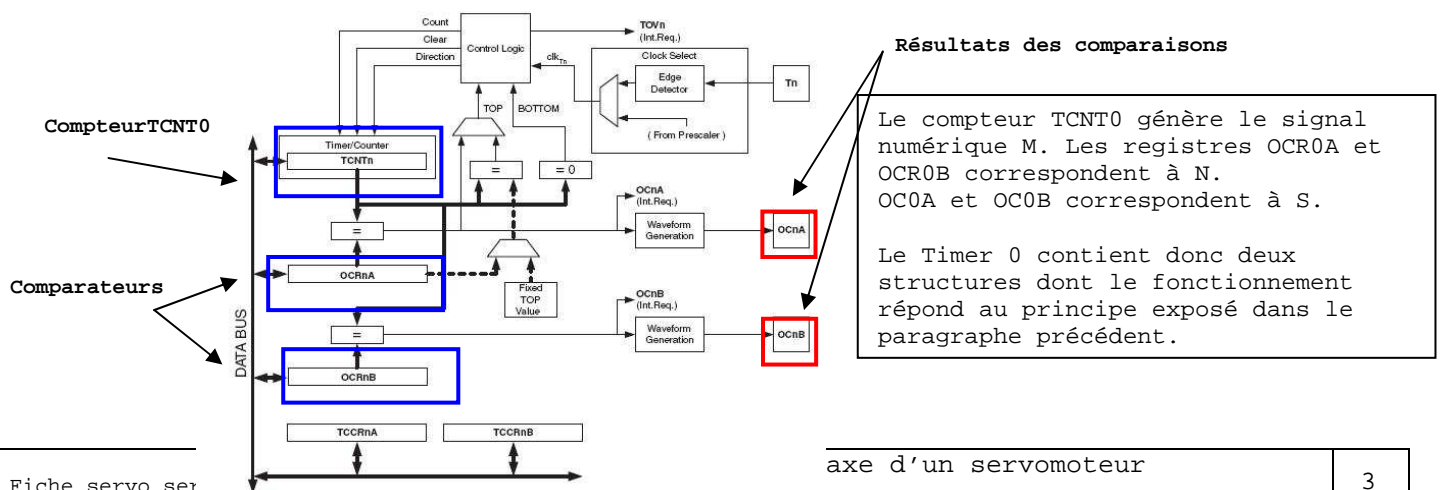
$$S_{moy} = \alpha \cdot S_{max}$$

#### • Génération d'un signal M.L.I. avec le microcontrôleur ATINY13

Le **Timer0** de l'ATINY13 permet de générer deux signaux modulés en largeur d'impulsion (OC0A, OC0B)

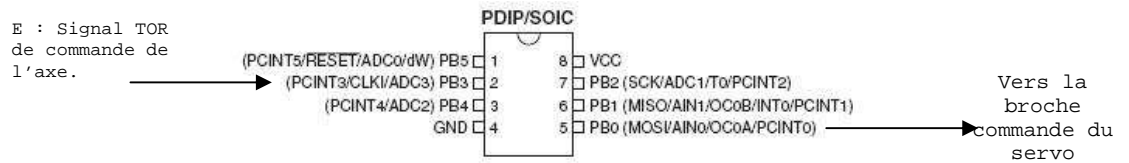
Le Timer0 intègre un **compteur (TCNT0)**, deux **comparateurs (OCROA, OCROB)** et divers registres. En mode M.L.I. son fonctionnement répond au principe exposé dans le paragraphe précédent.

Schéma fonctionnel du timer 0 de l'ATINY13



- **Commande du servo**

Le signal de commande du servomoteur (OC0A) est issu de la broche PB0 du microcontrôleur ATINY13. Ce signal commande directement le servomoteur.



La modification du rapport cyclique  $\alpha$  du signal OC0A se fait en modifiant la valeur contenue dans le registre OCR0A.

## B) Travail de préparation

Pour commander le servomoteur, vous devez réaliser un programme de réglage du rapport cyclique  $\alpha$  du signal OC0A en fonction de l'état logique de l'entrée PB3.

Ce travail peut être réalisé en plusieurs étapes :

Etape 1 : Détermination de l'expression de la valeur à placer dans le registre OCR0A en fonction du rapport cyclique souhaité.

Etape 2 : Calcul des paramètres nécessaires à la configuration du timer 0.

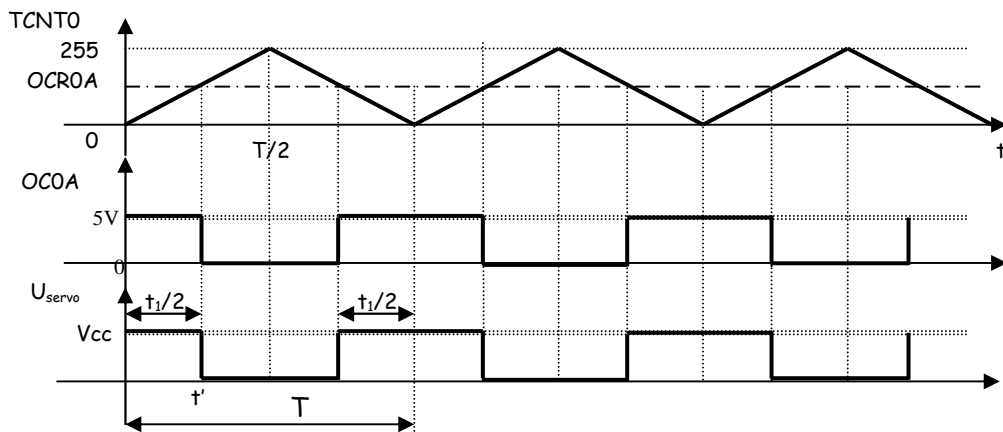
Etape 3 : Création d'un projet.

Etape 4 : Ecriture du programme.

Etape 5 : Programmation du composant, et tests.

### B1/ Etape 1 : Détermination de l'expression de la valeur à placer dans le registre OCR0A en fonction du rapport cyclique souhaité ( $OCR0A = f(\alpha)$ ).

On donne les chronogrammes ci-dessous :



a) Exprimez  $TCNT0 = f(t)$  pour  $t \in [0, T/2]$

b) Exprimez  $t' = f(t_1)$  (1)



c) A  $t = t'$ ,  $TCNT0 = OCR0A$ , exprimez  $t' = f(OCR0A)$  (2)

---

---

---

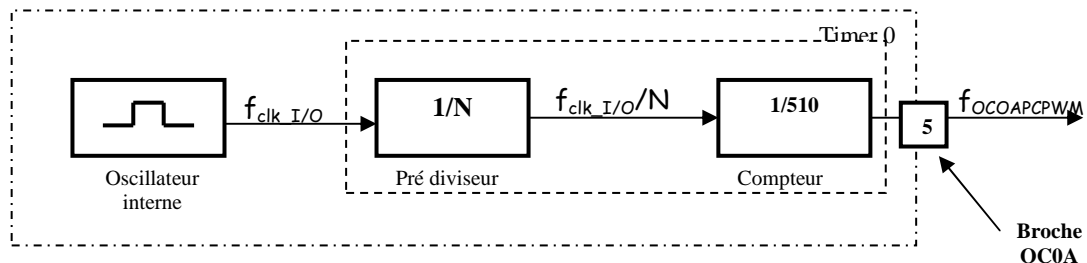
d) Exprimez  $OCR0A = f(\alpha)$  à partir des expressions (1) et (2)

---

---

## B2/ Etape 2 : Calcul des paramètres nécessaires à la configuration du timer 0.

On donne ci-dessous une représentation fonctionnelle du Timer 0 du microcontrôleur ATINY13.



Pour utiliser ce timer, il faut régler les valeurs de  $f_{clk\_I/O}$ ,  $f_{clk\_I/O/N}$  et le coefficient  $1/510$  dans le programme source en C. Pour l'application, ces valeurs doivent être choisies en fonction de la période du signal de commande du servomoteur.

### • Fréquence $f_{OCOAPCPWM}$ du signal OC0A

a) Quelle est la valeur maximum de la période ( $T_{OCOAPCPWM}$ ) du signal de commande d'un servomoteur ?

---

---

Le signal OC0A est obtenu à partir de la structure correspondant au schéma fonctionnel ci-dessus.

b) Exprimez  $f_{OCOAPCPWM}$  en fonction de  $f_{clk\_I/O}$ .

---

---

Dans notre application la fréquence  $f_{clk\_I/O} = 9,6\text{MHz}$ .

c) Choisissez la valeur de  $N$  parmi (1, 8, 64, 256 ou 1024) afin que  $4\text{ms} \leq T_{OCOAPCPWM} \leq 20\text{ms}$  (vous prendrez  $f_{OCOAPCPWM} = 50\text{Hz}$ ). Prendre la valeur de  $N$  la plus proche parmi celles proposées !

---

---



d) Calculez la valeur de  $f_{clk\_IO} / N$  et choisissez parmi (9600kHz, 1200kHz, 150kHz, 37.5kHz, 9.375kHz)

---



---

e) Calculez  $T_{OCOAPCPWM}$  pour la valeur choisie ci-dessus et vérifiez que cette valeur peut être utilisée par le servomoteur.

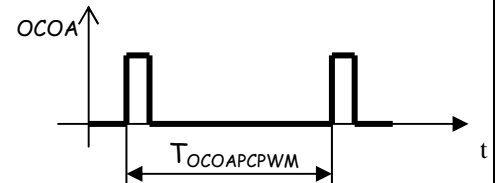
---



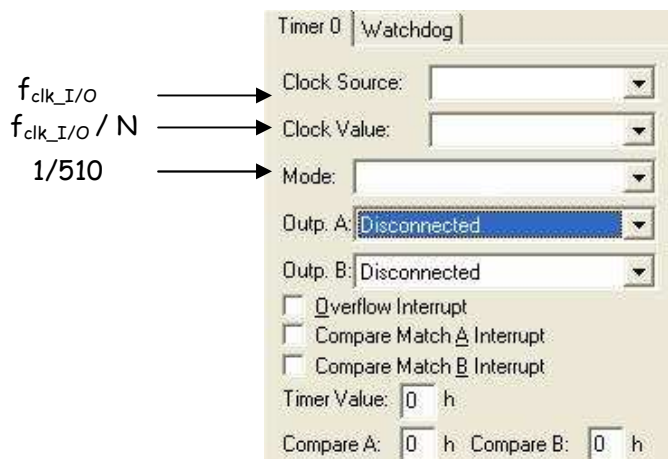
---



---



Vous disposez maintenant des paramètres nécessaires à la configuration du timer 0 de l'ATINY13.



Ceux-ci ( $f_{clk\_I/O}$ ,  $N$  et le coefficient  $1/510$ ) seront réglées dans le programme source en utilisant la boîte de dialogue du magicien (Wizard) du logiciel CodeVisionAvr ci-contre.

L'utilisation de cette boîte de dialogue est explicitée dans l'étape 3 ci-dessous.

### B3/ Etape 3 : Création d'un nouveau projet

Lancez le logiciel **CodeVisionAVR**

Dans la barre d'outils : « **File** » puis « **New** » pour obtenir la boîte de dialogue ci-contre.  
Cochez « **Project** » puis clic sur « **Ok** »



Ici « **Yes** »



- Sélection du composant cible

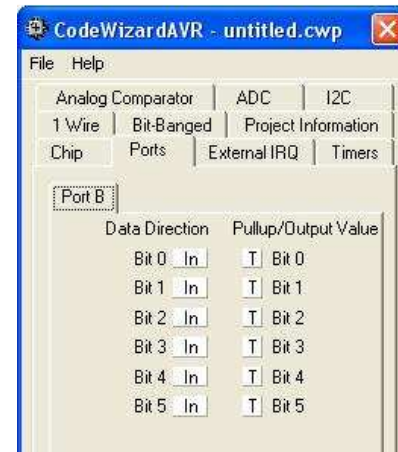


La boîte du « **Magicien** » s'ouvre comme ci-contre. Choisissez le « **Chip** » ATINY13 et réglez le signal d'horloge « **Clock** » à 9,6Mhz.

- Configuration du port

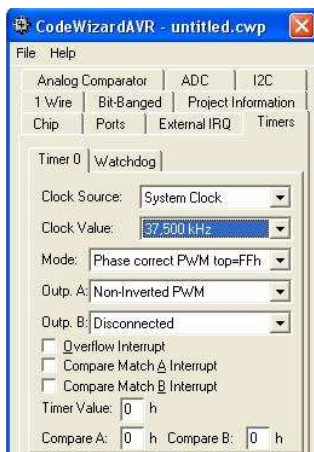
- Sélectionnez l'onglet « **Port B** ».

Configurez les bits 0 et 3 après avoir déterminé leur sens (entrée ou sortie). Sélectionnez P= 'Pullup' pour l'entrée.



- Configuration du timer 0

- Sélectionnez l'onglet « **Timer 0** ».



Configurez la boîte de dialogue comme ci-contre :

- Enregistrement du projet



- Sélectionnez « **Program Preview** ».



Si le projet est correctement configuré, les éléments suivants doivent se trouver dans le fichier source du programme.

```
#include <tiny13.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Crystal Oscillator division factor: 1
CLKPR=0x80;
CLKPR=0x00;

// Input/Output Ports initialization
// Port B initialization
// Func0=Out Func1=In Func2=In Func3=In Func4=In Func5=In
// State0=0 State1=T State2=T State3=P State4=T State5=T
PORTB=0x08;
DDRB=0x01;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 37,500 kHz
// Mode: Phase correct PWM top=FFh
// OC0A output: Non-Inverted PWM
// OC0B output: Disconnected
TCCR0A=0x81;
TCCR0B=0x04;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// External Interrupt(s) initialization
// INT0: Off
// Interrupt on any change on pins PCINT0-5: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Output: Off
ACSR=0x80;
ADCSRB=0x00;

while (1)
{
// Place your code here

};
}
```

Fermez la fenêtre « **Program Preview** ».

Sélectionnez « **Generate, save and Exit** ».

Donnez le nom **Servo** à votre projet. (Le compilateur demande trois fichiers, prendre trois fois le même nom)





#### B4/ Etape 4 : Ecriture du programme

Dans cette partie, vous allez compléter la **partie déclarative**

```
void main(void)
{
// Declare your local variables here
```

et la partie **exécutive** du programme.

```
while (1)
{
// Place your code here

};
```

On rappelle que la **partie déclarative** d'un programme est la zone dans laquelle sont **créées les variables** alors que la **partie exécutive** est la zone de **traitement** de ces variables.

##### (1) Partie exécutive du programme à réaliser

On désigne l'entrée PB3 par le terme « Choix\_Angle » et le registre OCROA par le terme « Trappe ». On définit deux constantes « Ouvert » et « Fermee ».

- a) En utilisant les termes « Choix\_Angle », « BRAS », « RENTRE » et « SORT » ci-dessus, écrivez en français les actions que doit réaliser le programme.

---

---

---

---

- b) Mettre le texte du a) sous la forme d'un algorithme ou d'un algorithme

- c) Voir prof pour le coder en langage C.

---

---

---

---

##### (2) Partie déclarative

Complétez le texte de votre programme comme ci-dessous

```
#include <tiny13.h>

#define Trappe      OCROA
#define Choix_Angle PINB.3
#define Ouvert      ?           // ? valeurs numériques à définir
#define Fermee      ?           // après avoir établi une relation entre
                                // l'angle  $\theta$  cherché et le rapport cyclique  $\alpha$ 
```



## Etape 5 : Programmation du composant, et test.

### C) Programmation du composant

#### Remarque importante concernant la programmation

Le composant doit être programmé et testé sur la carte SSI avec l'adaptateur ATINY13 / AT90S8535 avant d'être placé sur la carte « commande servomoteur ». (Annexe 1 et 2)

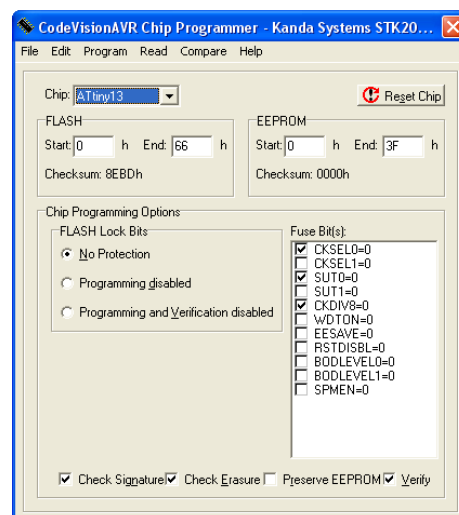
#### Configurez le projet

- > Project
- > Configure
  - > Sélectionnez l'onglet "After Make"
  - > Cochez "Program the Chip", CKSEL0=0, SUT0=0, CKDIV8=0.

La procédure de programmation est la suivante :

- Placez les 4 cavaliers de la carte sur la position « program » (Annexe 1)
- Compilez le projet (icône « compile project »),
- Créez le projet (icône « make project ») puis fermez la boîte de dialogue.
- Ouvrez la boîte de dialogue « run the chip programmer » et vérifiez que la cible est bien l'ATINY13 et que la zone « Fuse Bit(s) » est bien configurée comme ci-contre.
- Effacez le composant par :
  - program
  - Erase chip
- Vérifiez l'effacement par :
  - program
  - Blank Check
- Programmez le composant par :
  - program
  - flash
- Placez le cavalier I1 sur la position « Utilisation » (Annexe 1)

*Appel prof*



### D) Test du programme

Le signal MLI OC0A<sub>(ATINY13)</sub> est accessible sur la douille PWMA de la carte SSI.

La procédure de test est la suivante :

- 1) Connectez la douille PWMA de la carte SSI à une voie d'un oscilloscope.

Le rapport cyclique doit changer lorsque vous appuyez sur le BP « ENTR ».

- 2) Vérifiez la valeur de T<sub>OCOAPCPWM</sub> et les deux valeurs du rapport cyclique.

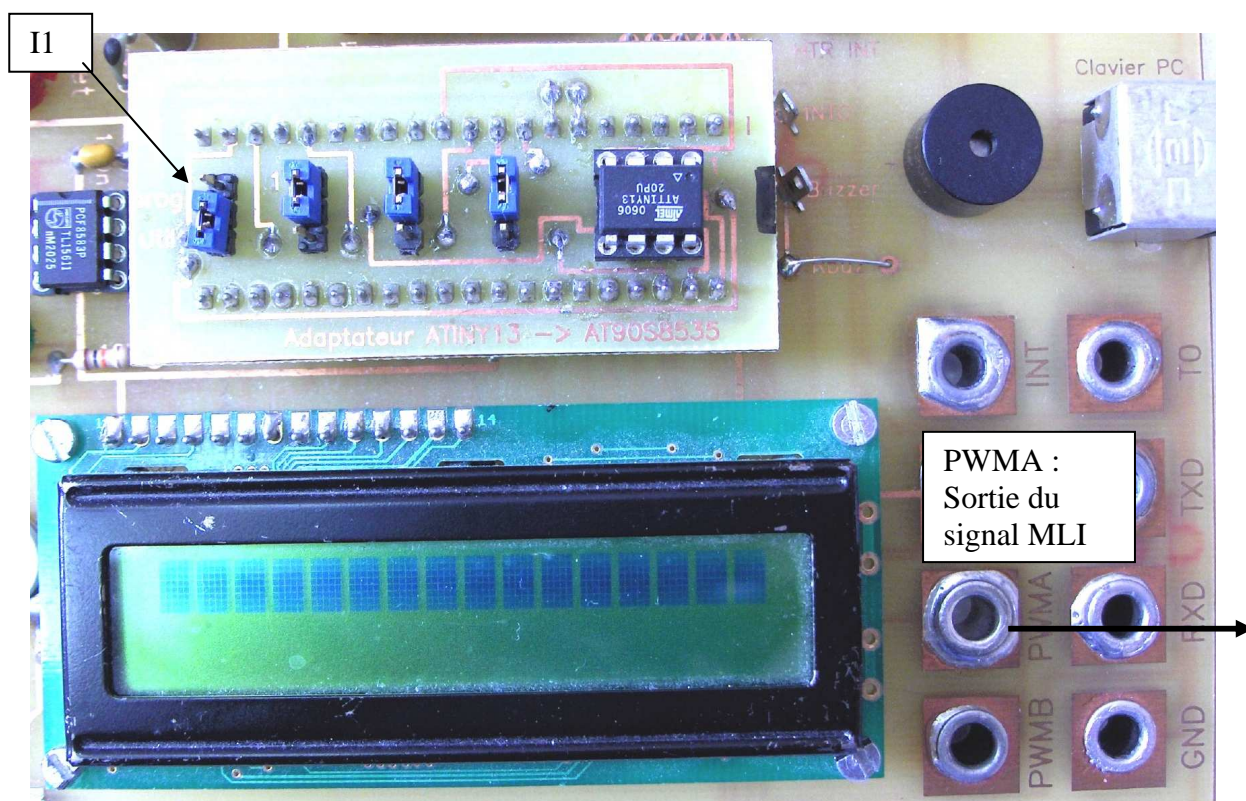
Si le fonctionnement est conforme à celui attendu :

- 3) Câbler un servomoteur à la carte SSI.
- 4) Ajustez expérimentalement les valeurs à placer dans OCR0A pour obtenir les angles recherchés.

*Appel prof*



## Annexe 1 : Utilisation de l'adaptateur ATINY13 / AT90MEGA8535



### Position des cavaliers

Modes	Position des cavaliers	I1	I2	I3	I4
Programmation		MOSI	MISO	SCK	Reset
Utilisation (ATINY13 -> SSI)		PB0 -> PD5	PB1 -> PD4	PB2 -> PD2	PB5 -> PA2

## Annexe 2 : Carte servomoteur

