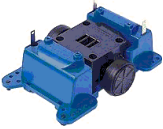




Fiche guide 5	TS SI		P.P.E. Robot suiveur de ligne	
Synthèse	4h			
	<h1>Commande d'un servomoteur</h1>			

Nom(s) :	Classe :	Groupe :
----------	----------	----------

Objectif

Commander la position de l'axe d'un servomoteur avec le timer d'un microcontrôleur.

Matériels

Carte ATMESSSI V1 + Module interface ATINYxx pour SSI + Carte ATINYxx + Alimentation 10V.
Servomoteur Futaba 93003.

Logiciel : CodeVisionAvr.

Documentation

Schémas des différentes cartes.

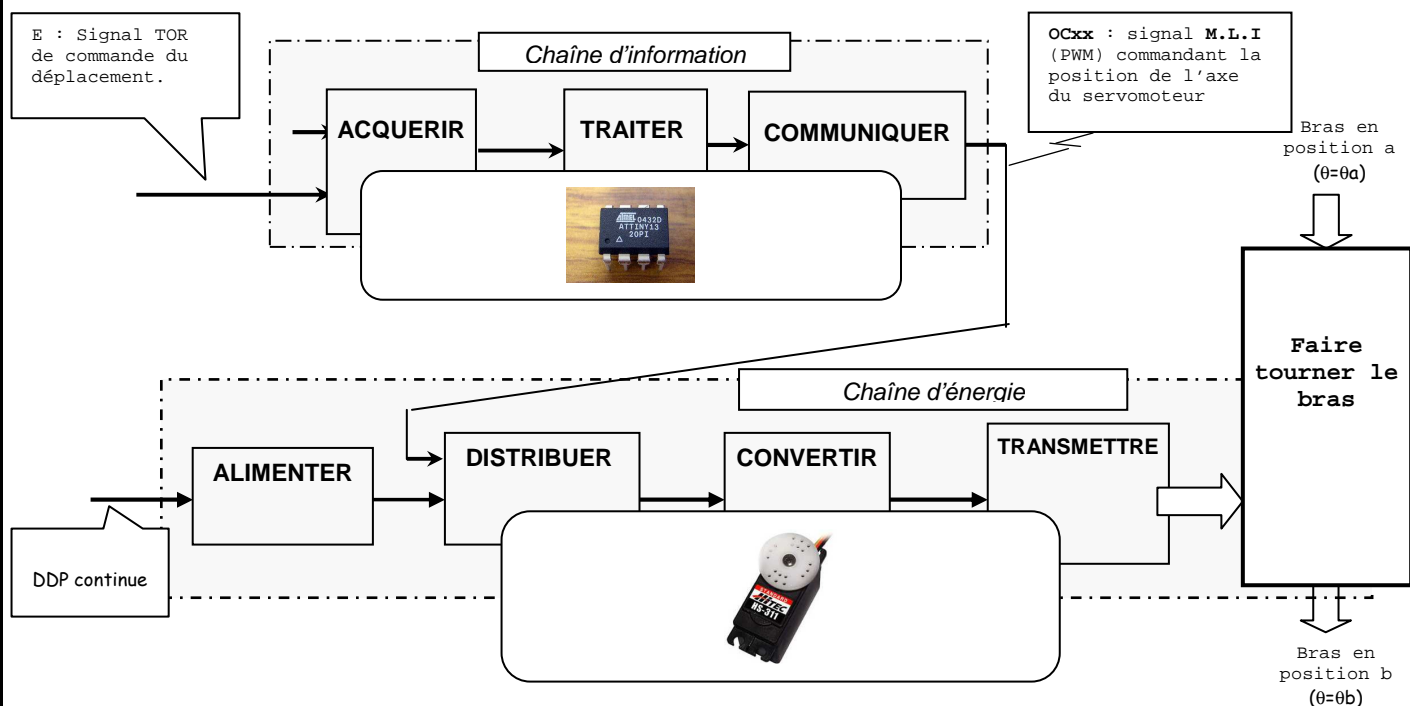
Le présent document et les schémas sont téléchargeables sur le site WebGE à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)

Infos sur les servomoteurs : http://fribotte.free.fr/bdtech/pic/pic_et_servo.html

A) Présentation

A1) Schéma fonctionnel

La rotation du bras du robot peut être réalisée avec un servomoteur. Le présent document doit vous permettre de commander un tel dispositif avec le timer d'un μC .



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

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

Votre travail va se limiter à la **réalisation du logiciel** à implanter dans le microcontrôleur.

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 de régler la position de l'axe d'un servomoteur.

Dans un premier temps, vous allez rechercher les caractéristiques du signal de commande d'un servomoteur. Puis, à partir de la **résolution angulaire souhaitée** et des caractéristiques du servomoteur utilisé, vous choisirez le microcontrôleur à utiliser. Enfin, vous calculerez les différents paramètres nécessaires au réglage du timer du microcontrôleur.

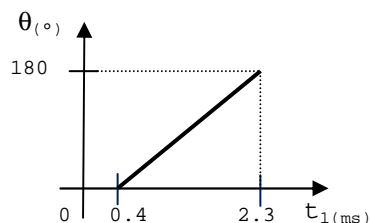
A2) Caractéristiques du signal de commande d'un servomoteur

Consultez le site cité en page de garde.

A21) Dessinez, ci-dessous, le signal de commande d'un servomoteur, **précisez** ses caractéristiques (fréquence ou période maximum, tension, rapport cyclique α ...)

Ce type de signal est dit modulé en largeur d'impulsions (MLI ou PWM). Vous allez découvrir le moyen de produire ce type de signal dans le paragraphe A3.

Des mesures faites sur le servomoteur utilisé ont permis de représenter sa position angulaire en fonction de t_1 soit $\theta_{(^\circ)} = f(t_1)$. t_1 est le temps à l'état haut du signal de commande.



A22) Donnez l'expression de $\theta = f(t_1)$ pour $t_1 \in [0,4;2,3]$. Cette équation sera réutilisée lors du choix du timer.



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

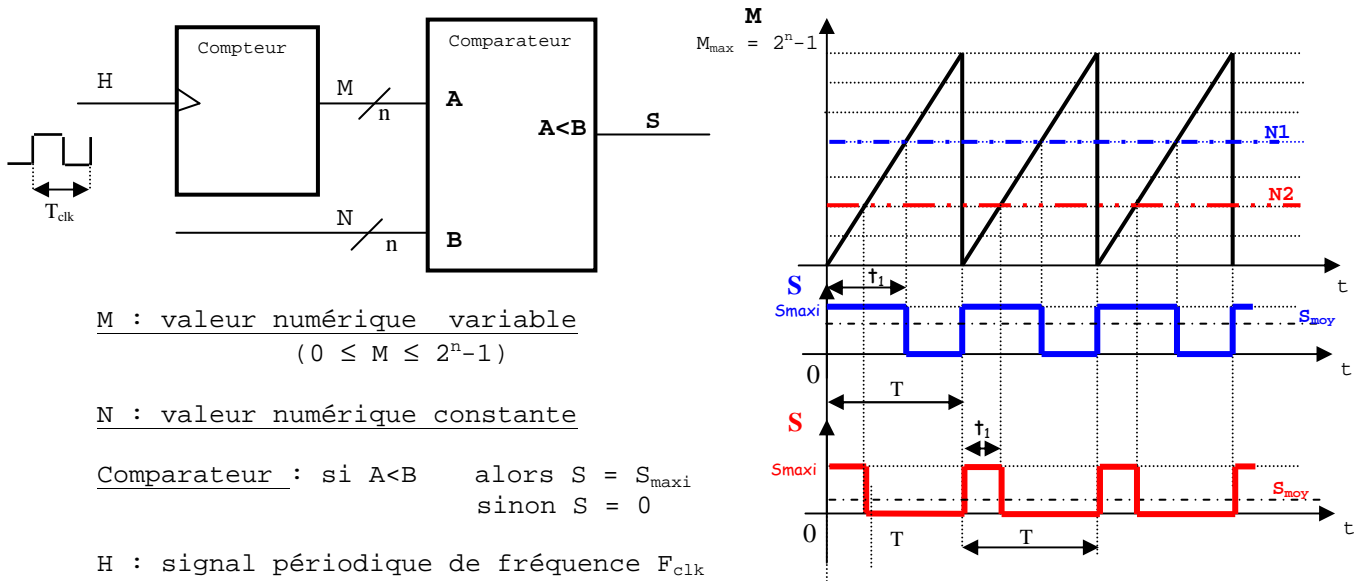
• Principe

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 α 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 ($N1$, $N2$).



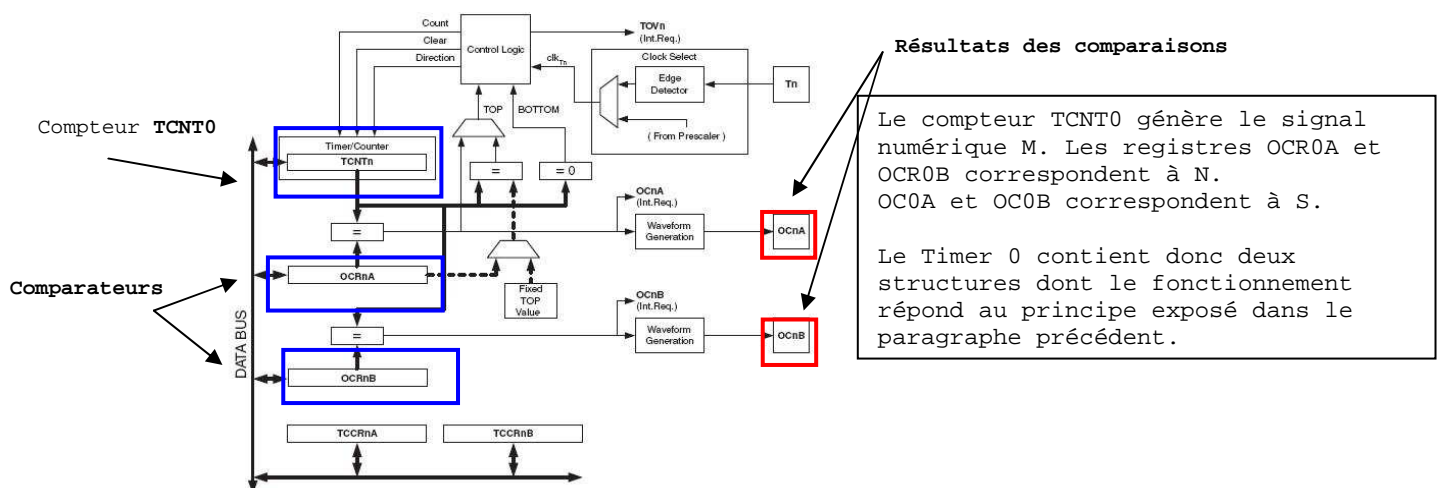
Dans les microcontrôleurs, les signaux modulés en largeur d'impulsion sont générés par une structure appelée **TIMER**. Celle-ci répond au principe développé ci-dessus.

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

• Exemple : génération d'un signal M.L.I. avec le microcontrôleur ATINY13

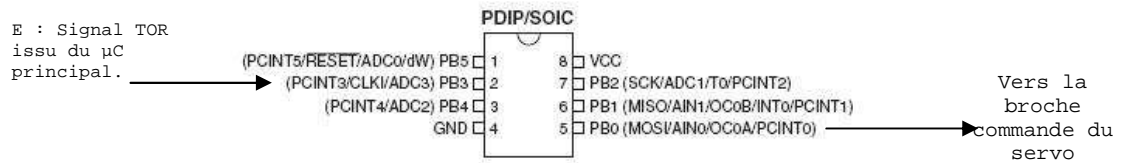
Le **Timer0** de l'ATINY13 permet de générer deux signaux modulés en largeur d'impulsion (**OC0A**, **OC0B**). Ce timer intègre un **compteur (TCNT0)**, deux **comparateurs (OCROA, OCROB)** et divers registres. En mode M.L.I. son fonctionnement répond au principe exposé dans le paragraphe précédent.

o Schéma fonctionnel du timer 0 de l'ATINY13 (documentation constructeur)



- **Exemple de commande d'un servomoteur**

Si on utilise un ATINY13 pour commander un servomoteur, il suffit de relier le signal (**OC0A**), issu de la broche PB0 du microcontrôleur ATINY13, à l'entrée « signal de commande » (fil blanc) du servomoteur.



La modification du rapport cyclique α du signal **OC0A** se fait en modifiant la valeur contenue dans un registre nommé **OCR0A**.

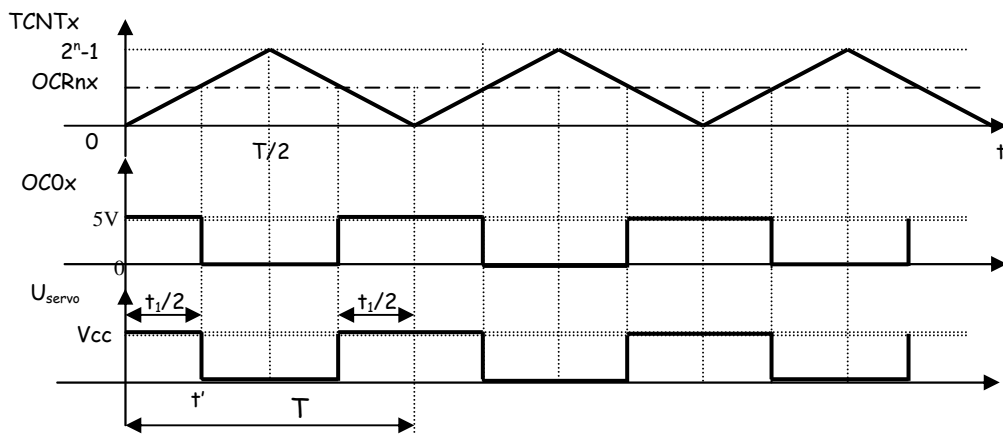
B) Travail de préparation

B1) Etape 1 : choix du timer du microcontrôleur

Le choix du timer du microcontrôleur doit se faire en fonction de la **résolution angulaire** r_θ (angle de déplacement minimum) souhaitée et de la caractéristique $\theta(t_1)$ du servomoteur utilisé (caractéristique déterminée en A22). Pour cela, vous allez établir une relation entre la résolution angulaire r_θ , la fréquence F du signal PWM et le registre de comparaison du timer. Nous appellerons ce registre **OCRnx**.

B11) Détermination de l'expression de la valeur à placer dans le registre **OCRnx** en fonction du rapport cyclique souhaité ($OCRnx = f(\alpha)$)

On donne les chronogrammes ci-dessous :



B111) Exprimez $TCNTx = f(t)$ pour $t \in [0, T/2]$

B112) Exprimez $t' = f(t_1)$ (1)

B113) A l'instant $t = t'$, $TCNTx = OCRnx$. **Exprimez** $t' = f(OCRnx)$ (2)

B114) Exprimez $OCR_{nx} = f(\alpha)$ à partir des expressions (1) et (2).

B12) Expression de la résolution r_θ en fonction de OCR_{nx} et de F

Le timer du microcontrôleur règle α par pas $p = 1/(2^n - 1)$.

n dépend du timer utilisé ($n = 8, 9$ ou 10) selon le microcontrôleur utilisé.

B121) Exprimez t_1 en fonction de OCR_{nx} et de la fréquence F du signal de commande.

B122) A partir de l'expression précédente et de celle déterminée au paragraphe A22, **déterminez** θ en fonction de OCR_{nx} et de la fréquence F du signal de commande.

Pour $t_1 \in [0, 4; 2, 3]$, on peut écrire que $\Delta\theta = [94,7 \cdot 10^3 \cdot \Delta OCR_{nx}] / [(2^n - 1)F]$. La résolution angulaire r_θ peut être déterminée pour $\Delta OCR_{nx} = 1$.
Soit **$r_\theta = 94,7 \cdot 10^3 / ((2^n - 1)F)$** .

On souhaite obtenir une **résolution angulaire $r_\theta \leq 2,5^\circ$**

B123) Déterminez un encadrement pour n sachant que $50 \leq F_{(Hz)} \leq 250$

B124) Choisissez un timer et un microcontrôleur parmi ceux proposés ci-dessous :

	n	
	Timer 0	Timer 1
ATINY13 (1,86€)	8	
ATINY24 (2,16€)	8	8 ou 9 ou 10

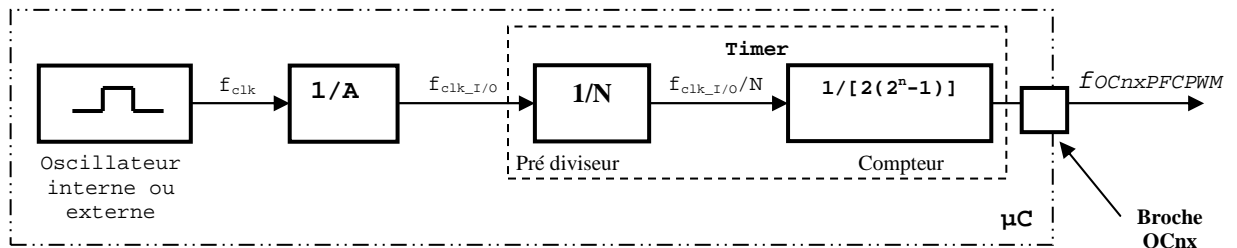
B13) Choix de la fréquence F

B131) Calculez la valeur de la fréquence F à partir de la valeur de n choisie précédemment.

Le paragraphe suivant va vous permettre de définir la valeur de réglage de la fréquence F délivrée par le timer.

B2/ Etape2 : calcul des paramètres nécessaires à la configuration du timer.

On donne ci-dessous une représentation fonctionnelle d'un timer (ATINY13, ATINY24).



A est réglable et peut prendre les valeurs 1, 2, 4, 8, 16, 32, 64, 128 ou 256. N est réglable et peut prendre les valeurs 1, 8, 64, 256 ou 1024 avec le microcontrôleur utilisé.

Pour utiliser le timer, il faut régler les valeurs des fréquences f_{clk} , $f_{clk_I/O}$, $f_{clk_I/O}/N$ et le coefficient $1/[2(2^n-1)]$ dans le fichier source du programme.

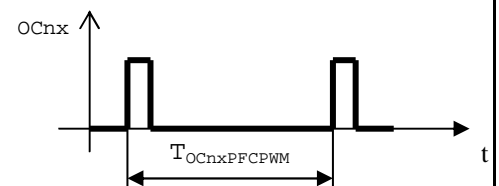
La **fréquence** $f_{OCnxPFCPWM}$ correspond à la fréquence F calculée précédemment.

B21) Calculez $f_{clk_I/O}/N$.

La fréquence f_{clk} du signal de l'oscillateur peut être produite avec un quartz externe (pour l'ATINY24) ou avec l'oscillateur interne du microcontrôleur (4,8MHz ou 9,6MHz pour l'ATINY13, 8MHz pour l'ATINY24). La première solution doit être préférée lorsqu'on souhaite une grande précision.

B22) Peut-on utiliser l'oscillateur interne du microcontrôleur pour produire $f_{clk_I/O}/N$?

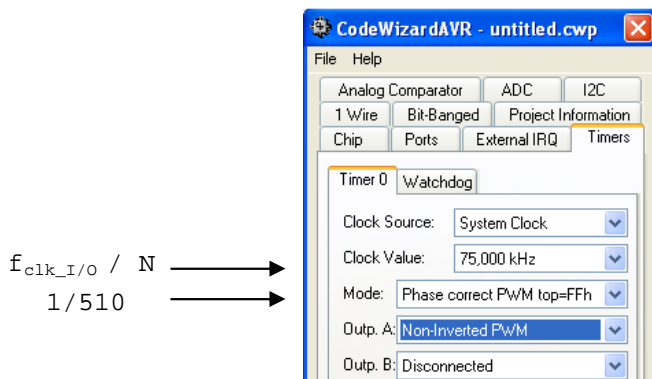
Pour répondre à cette question, il faut choisir un produit $A.N$ et vérifier que la fréquence f_{clk} et le produit $A.N$ choisis sont compatibles avec r_0 et $f_{OCnxPFCPWM}$.



Si l'oscillateur interne n'est pas utilisable, **calculez** la fréquence du quartz à placer à l'extérieur du composant et **choisissez** celui qui s'en rapproche le plus dans le tableau de l'annexe 3.

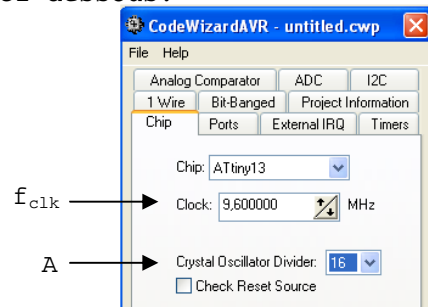
Vous disposez maintenant des paramètres nécessaires à la configuration du timer n de l'ATINYxx.

Exemple pour un ATNIY13



Ces paramètres (f_{clk} , A , $f_{clk_I/O} / N$ et le coefficient $1/510$) sont réglées dans le programme source en utilisant les boîtes de dialogue du magicien (Wizard) du logiciel CodeVisionAvr ci-contre.

L'utilisation de ces boîtes de dialogue est expliquée dans l'étape 3 ci-dessous.



B3/ Etape 3 : Création d'un nouveau projet

La démarche ci-dessous est donnée pour un ATNIY13. Voir le prof si un autre μC est utilisé.

Lancez le logiciel CodeVisionAVR

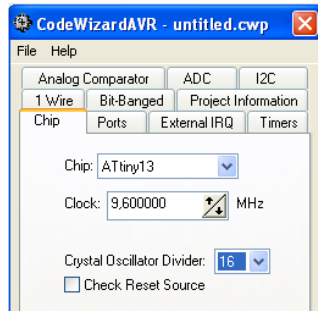
Dans la barre d'outils : « **File** » puis « **New** » pour obtenir la boîte de dialogue ci-contre. Cochez « **Project** » puis clic sur « **Ok** »



Ici « **Yes** »



- Sélection du composant cible



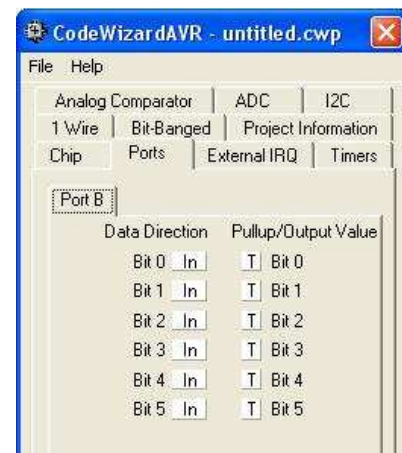
La boîte du « **Magicien** » s'ouvre comme ci-contre. Choisissez le « **Chip** » ATINY13 et réglez le signal d'horloge « **Clock** » et le facteur de division « **A** » avec la valeur déterminée en B22.

- Configuration du port B

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

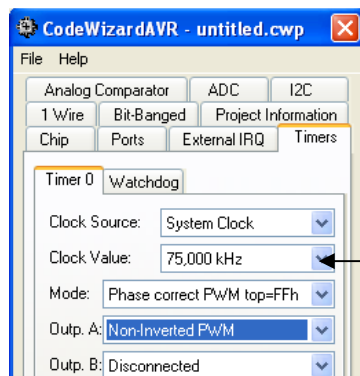
Configurez les bits 0 et 3 après avoir déterminé leur sens (entrée ou sortie).

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: 16
    CLKPR=0x80;
    CLKPR=0x04;

    // 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: 75,000 kHz
    // Mode: Phase correct PWM top=FFh
    // OC0A output: Non-Inverted PWM
    // OC0B output: Disconnected
    TCCR0A=0x81;
    TCCR0B=0x02;
    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 appelle « Choix_Angle » l'entrée TOR PB3 et « Bras » le registre OCROA. On définit deux constantes « RENTRE » et « SORT ».

B41) En utilisant les termes « Choix_Angle », « Bras », « RENTRE » et « SORT », **écrivez** l'algorithme ou dessinez l'algorithme du programme de commande du bras.

B42) Codez cet algorithme (algorithme) en langage C.



B43) Exprimez $OCR0A = f(\theta)$ à partir des résultats de l'étape 1.

B44) Complétez le tableau ci-dessous

$\theta(^{\circ})$	0	45	90	135	180
OCR0A					

(2) Partie déclarative

Complétez le texte de votre programme comme ci-dessous

```
#include <tiny13.h>

#define BRAS OCR0A
#define Choix_Angle PINB.3      // Rentre = 0°      SORT = 90°
#define RENTRE      ?          // ? valeurs numériques à définir
#define SORT        ?          // avec l'expression du B44)
```



C) Programmation du composant

Remarque importante concernant la programmation

Le composant doit être programmé et testé sur la carte SSI avec l'adaptateur ATINY13 / ATMEGA8535 avant d'être placé sur la carte « commande servomoteur ». (Annexe 1 et 2)

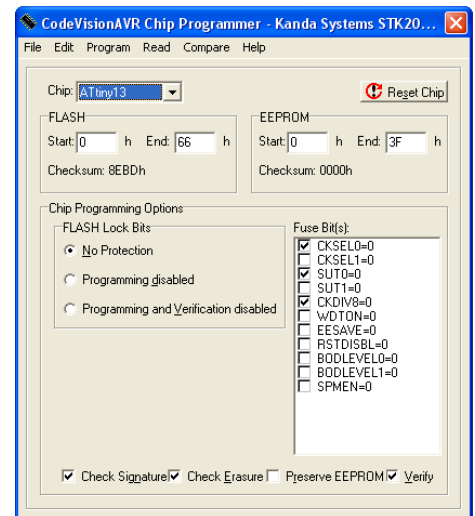
Configurez le projet

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

Appel prof

La procédure de programmation est la suivante :

- **Placez** les 4 cavaliers de la carte sur la position « prog » (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
 - Blanck Check
- **Programmez** le composant par :
 - program
 - flash
- **Placez** le cavalier I1 sur « Utilisation » (Annexe 1)



D) Test du programme

Le signal MLI OC0A_(ATINY13) est accessible sur la douille PWMA de la carte SSI.

La procédure de test est la suivante :

D1) 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 ».

D2) Vérifiez la valeur de $T_{OCOAPCPWM}$ et les deux valeurs du rapport cyclique.

Si le fonctionnement est conforme à celui attendu :

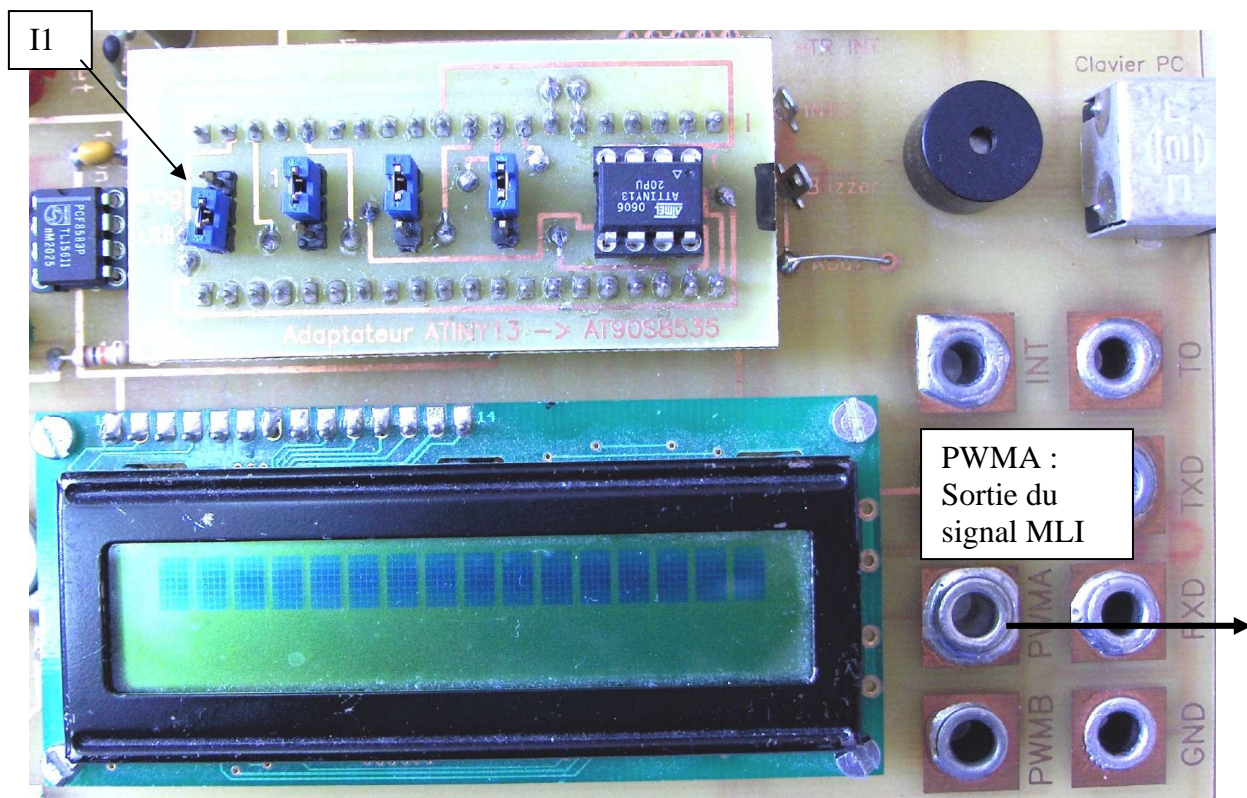
D3) Câblez un servomoteur à la carte SSI.

D4) Ajustez expérimentalement les valeurs à placer dans OCR0A pour obtenir les angles recherchés.

Appel prof



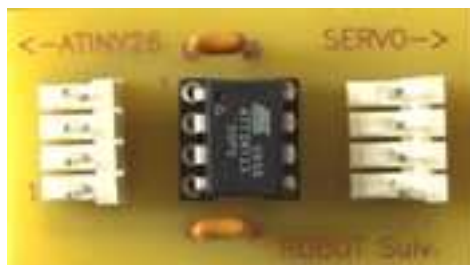
Annexe 1 : Utilisation de l'adaptateur ATINY13 / ATMEGA8535



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



HC49/4H CRYSTALS

Frequency	Holder	Specification	Stock No.	Alpha Code
3.27680MHz	HC49/4H	30/50/10/12	XTAL003052	A118C
3.579545MHz	HC49/4H	30/50/20/20	XTAL003063	A119K
3.68640MHz	HC49/4H	30/50/20/30	XTAL003263	A169K
4.0MHz	HC49/4H	20/50/10/30	XTAL003074	A120K
4.0320MHz	HC49/4H	30/50/20/30	XTAL003081	A121K
4.0960MHz	HC49/4H	30/50/10/30	XTAL003084	A122K
4.194304MHz	HC49/4H	30/50/10/30	XTAL003092	A123J
4.194304MHz	HC49/4H	30/50/10/12	XTAL003093	A123K
4.433619MHz	HC49/4H	30/50/10/20	XTAL003102	A124K
4.91520MHz	HC49/4H	30/50/20/30	XTAL003115	A127K
5.0MHz	HC49/4H	30/50/10/30	XTAL003119	A128K
5.760MHz	HC49/4H	30/50/10/30	XTAL003510	L102K
6.0MHz	HC49/4H	30/50/10/30	XTAL003132	A132K
6.1440MHz	HC49/4H	30/50/10/30	XTAL003137	A133K
7.37280MHz	HC49/4H	30/50/10/30	XTAL003335	A194K
7.37280MHz	HC49/4H	15/30/10/18	XTAL003336	A194L
7.864320MHz	HC49/4H	30/50/10/30	XTAL003145	A139A
8.0MHz	HC49/4H	30/50/20/30	XTAL003156	A140K
8.1920MHz	HC49/4H	30/50/10/30	XTAL003271	A170K
9.83040MHz	HC49/4H	30/50/10/30	XTAL003279	A173K
10.0MHz	HC49/4H	30/50/20/30	XTAL003169	A143K
10.7520MHz	HC49/4H	30/50/10/30	XTAL003365	A212K
11.05920MHz	HC49/4H	30/50/20/30	XTAL003523	L108K
12.0MHz	HC49/4H	30/50/20/30	XTAL003215	A158K
12.2880MHz	HC49/4H	30/50/10/30	XTAL003286	A175K
14.318180MHz	HC49/4H	30/50/20/30	XTAL003200	A153L
14.74560MHz	HC49/4H	30/50/10/30	XTAL003224	A159K

Frequency	Holder	Specification	Stock No.	Alpha Code
15.0MHz	HC49/4H	30/50/10/30	XTAL003236	A160K
15.360MHz	HC49/4H	20/30/10/30	XTAL003554	M451K
16.0MHz	HC49/4H	30/50/20/30	XTAL003240	A161K
16.93440MHz	HC49/4H	30/50/10/30	XTAL003366	A213K
18.4320MHz	HC49/4H	30/50/20/30	XTAL003176	A146K
19.66080MHz	HC49/4H	30/50/10/30	XTAL003309	A182K
20.0MHz	HC49/4H	30/50/20/12	XTAL003185	A147K
20.0MHz	HC49/4H	30/50/10/20	XTAL003186	A147L
24.0MHz	HC49/4H	30/50/10/30 Fund	XTAL003325	A189K
24.5760MHz	HC49/4H	30/50/10/20 Fund	XTAL003046	A116K
32.0MHz	HC49/4H	30/50/20/SR 3rd	XTAL003254	A166K
35.25120MHz	HC49/4H	15/30/20/18 3rd	XTAL003371	A216K
40.320MHz	HC49/4H	30/50/10/18 3rd	XTAL003379	A220H

