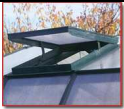




<b>Fiche élève :</b>  <b>MCC et PWM</b>	<b>TS SI</b>		<b>P.P.E Mini serre</b>	
	<b>Commande d'un moteur à courant continu en mode PWM</b>			

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

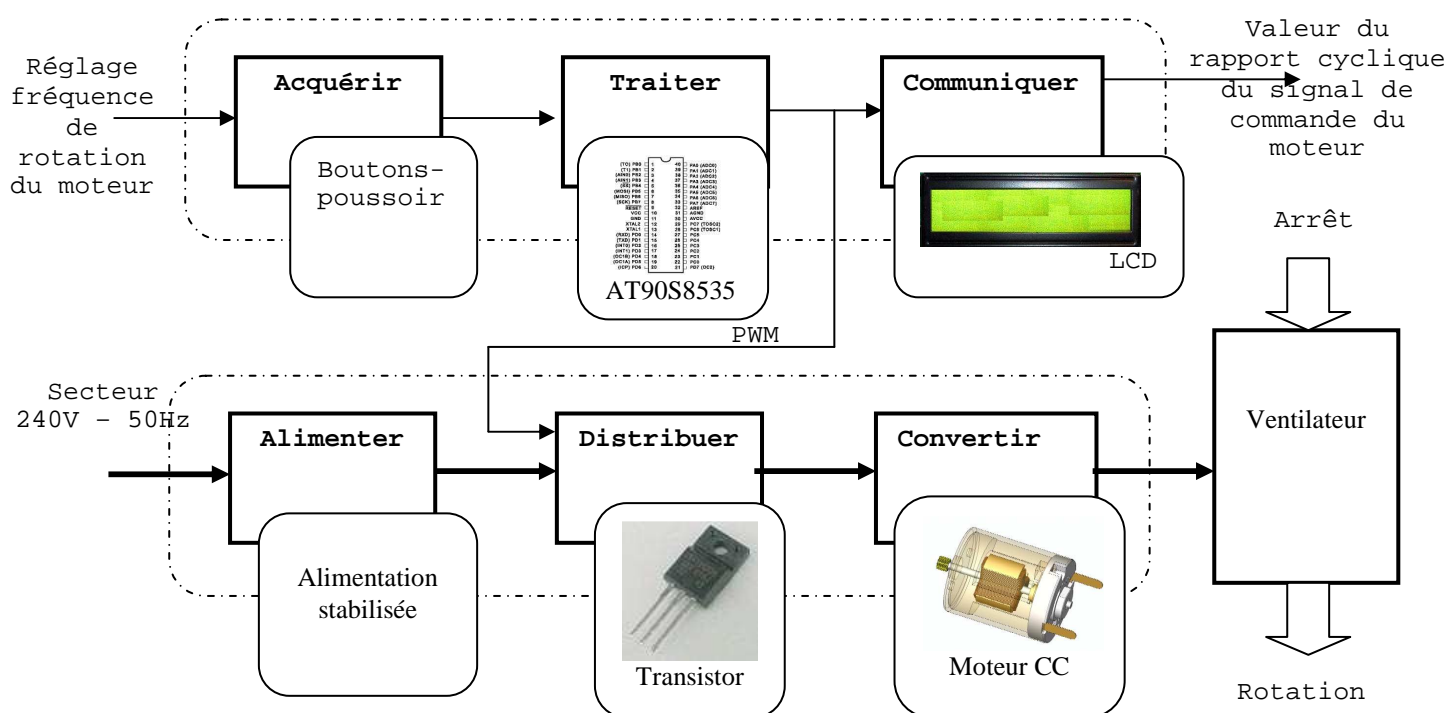
**Objectif :** Faire varier la fréquence de rotation du moteur du ventilateur.

<b>Matériels</b> Carte ATMESSSI V1 + Module interface de puissance + Alimentation 10V.
<b>Logiciels</b> CodeVisionAvr.
Sur le site WebGE à l'adresse <a href="http://p.mariano.free.fr/">http://p.mariano.free.fr/</a> (rubrique PPE) Schémas de la carte ATMESSSI V1 et de la carte interface de puissance. Documentation technique de l'afficheur LCD à processeur Hitachi.

## A) Présentation

### A1) Schéma fonctionnel

On souhaite faire varier la fréquence de rotation du moteur de ventilateur de la serre. Pour cela, on se propose de mettre en œuvre une structure correspondant au schéma ci-dessous.



La fonction « Traiter » est assurée par un programme implanté dans le microcontrôleur. Le signal PWM est issu d'une structure appelée « Timer ». Celle-ci est intégrée au microcontrôleur ATMEL. L'énergie du signal PWM n'étant pas suffisante pour entraîner le moteur à courant continu du ventilateur, la fonction « Distribuer » se charge de l'adapter. Cette fonction est réalisée par un transistor MOS. La fonction « Communiquer » est remplie par un afficheur LCD (processeur Hitachi).

L'ensemble des structures matérielles étant réunies sur la carte « ATMELSSI » et la carte « Interface de puissance », 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 moteur et de l'afficheur.

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 fréquence de rotation du moteur du ventilateur.

## A2) Principe de réglage de la fréquence de rotation d'un moteur à courant continu

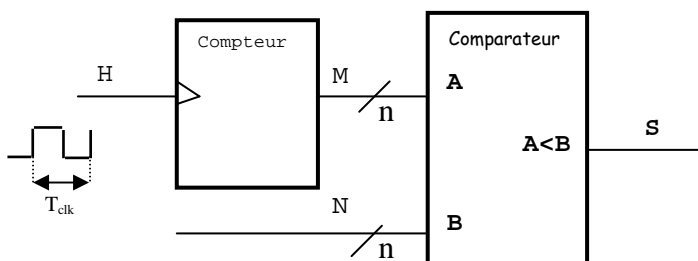
Le ventilateur est équipé d'un moteur à courant continu.

Pour faire varier la vitesse de rotation d'un moteur à courant continu, il suffit de modifier la valeur moyenne de la d.d.p. présente à ses bornes en le commandant par un signal dit « **MLI (ou PWM)** ».

- Génération d'un signal **Modulé en Largeur d'Impulsion** (principe)

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  $N$ .

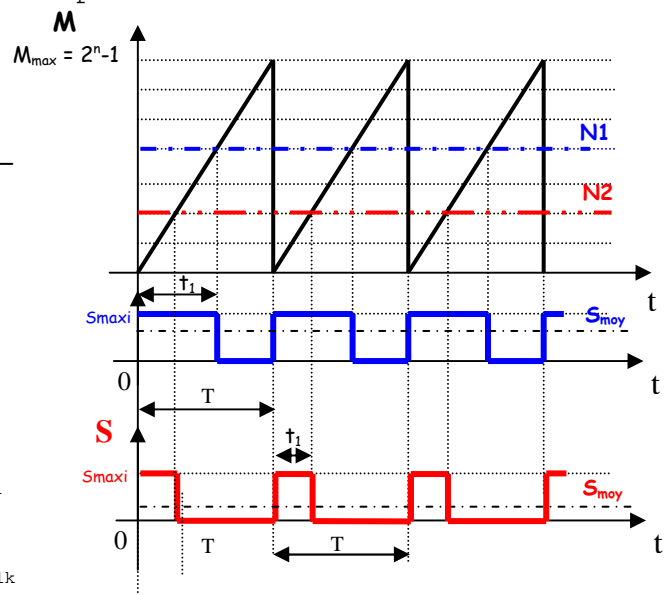


$M$  : valeur numérique variable  
( $0 \leq M \leq 2^n - 1$ )

$N$  : valeur numérique constante

Comparateur : si  $A < B$  alors  $S = S_{maxi}$   
sinon  $S = 0$

$H$  : signal périodique de fréquence  $F_{clk}$



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.

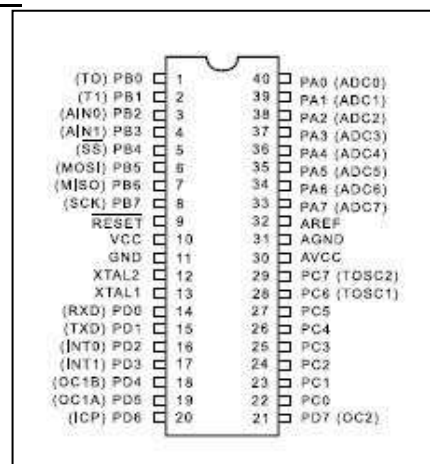
$$S_{moy} = \alpha \cdot S_{maxi}$$

\*M.L.I : Modulation de largeur d'impulsion (P.W.M. : Pulse With Modulation)

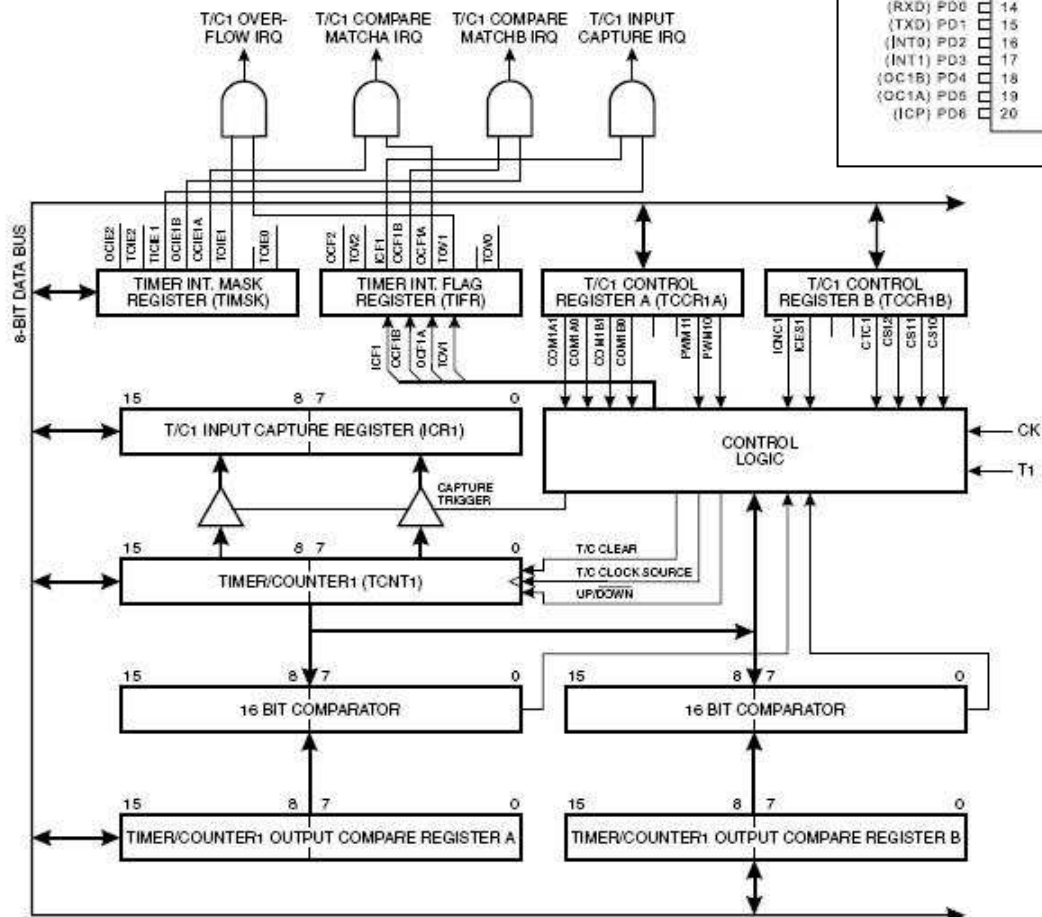
- Génération d'un signal MLI avec le microcontrôleur AT90s8535

Le **Timer 1** de l'AT90s8535 permet de générer deux signaux modulés en largeur d'impulsion. Ces signaux sont identifiés OC1B et OC1A. Ils sont accessibles sur les broches 4 et 5 du port D. (voir ci-contre)

Le Timer 1 intègre un **compteur**, des **comparateurs** et divers registres. En mode M.L.I. son fonctionnement répond au principe exposé dans le paragraphe précédent.



## Schéma fonctionnel du timer 1 de l'AT90S8535



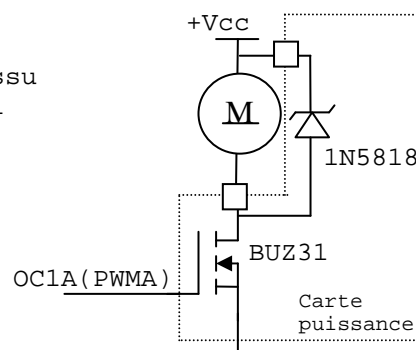
Le compteur TCNT1 génère le signal numérique M. Les registres OCR1A et OCR1B correspondent à N. Les broches OC1A et OC1B correspondent à S. Le Timer 1 contient donc deux structures dont le fonctionnement répond au principe exposé dans le paragraphe précédent.

La modification du rapport cyclique  $\alpha$  du signal de commande du moteur du ventilateur se fait en modifiant la valeur contenue dans le registre OCR1A.

- Amplification du signal PWM

Le signal de commande du moteur du ventilateur est issu de la broche PD5 (OC1A) du microcontrôleur. Ce signal est repéré PWMA sur la carte SSI.

L'énergie du signal PWM n'étant pas suffisante pour entraîner le moteur à courant continu du ventilateur, la fonction « Distribuer » (carte de puissance représentée ci-contre) se charge de l'adapter.

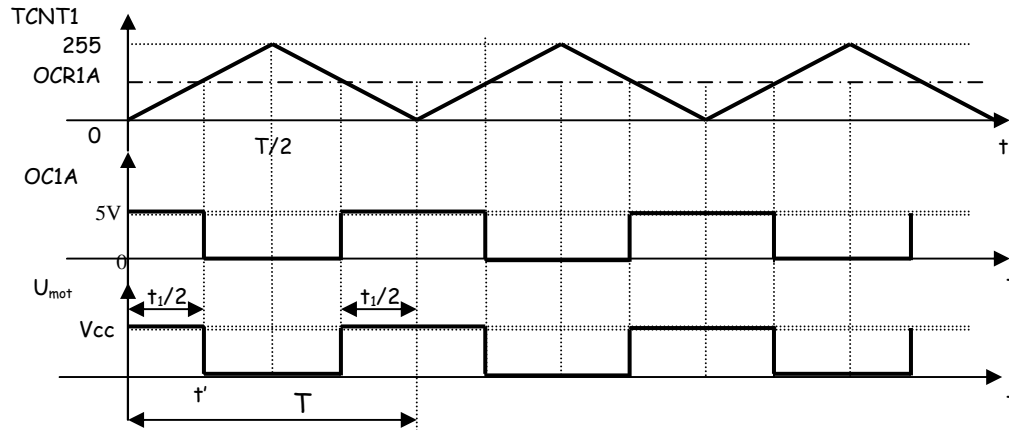


## B) Travail demandé

### B1/ Etape 1 : Détermination des valeurs de OCR1A en fonction de $\alpha$

**Objectif** : Déterminer les valeurs à placer dans le registre OCR1A pour régler la valeur du rapport cyclique  $\alpha$  du signal de commande du moteur du ventilateur.

On donne les chronogrammes ci-dessous :



Q1a) Exprimez  $TCNT1 = f(t)$  pour  $t \in [0, T/2]$

---

Q1b) Exprimez  $t' = f(t_1)$  (1)

---

Q1c) A  $t = t'$ ,  $TCNT1 = OCR1A$ , exprimez  $t' = f(OCR1A)$  (2)

---



---



---

Q1d) Exprimez  $OCR1A = f(\alpha)$  à partir des expressions (1) et (2)

---



---



---

Q1e) Complétez le tableau ci-dessous (arrondissez à l'entier supérieur)

$\alpha(\%)$	0	10	20	30	40	50	60	70	80	90	100
OCR1A											

## B2/ Etape 2 : Création et configuration d'un projet

Lancez le logiciel **CodeVisionAVR**

- **Création d'un nouveau projet**

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-dessous. Choisissez le « **Chip** » AT90S8535 et réglez le signal d'horloge « **Clock** » à 4Mhz.



- **Configuration des ports A et D**

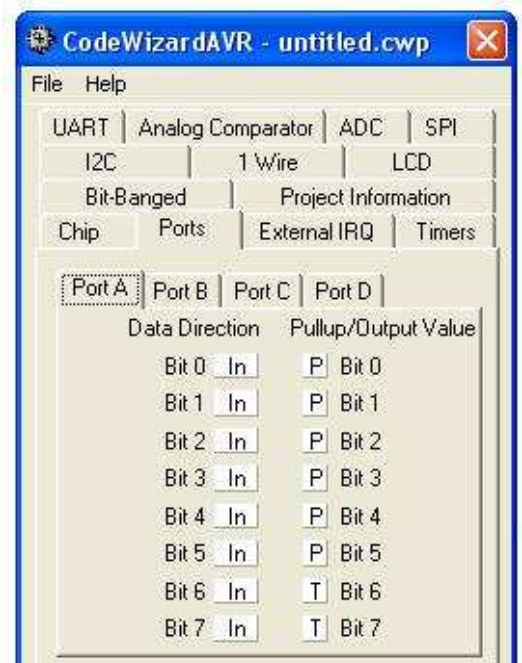
- Sélectionnez l'onglet « **Port A** ».

Les boutons-poussoirs de la carte ATMEL SSI sont connectés au Port A du microcontrôleur.  
(voir schéma structurel « **ATMELSSI V1** »)

Configurez le Port A comme ci-contre.

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

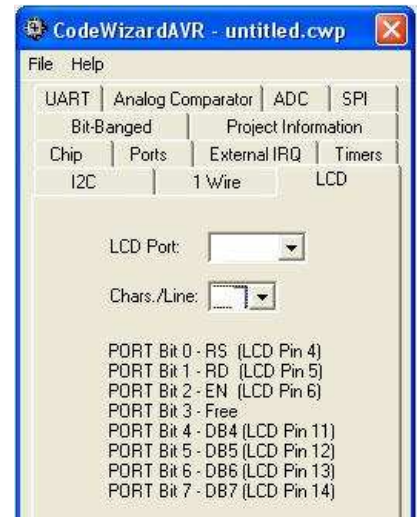
Configurez le bit 5 du port D après avoir déterminer son sens (entrée ou sortie).



- Choix de l'affichage

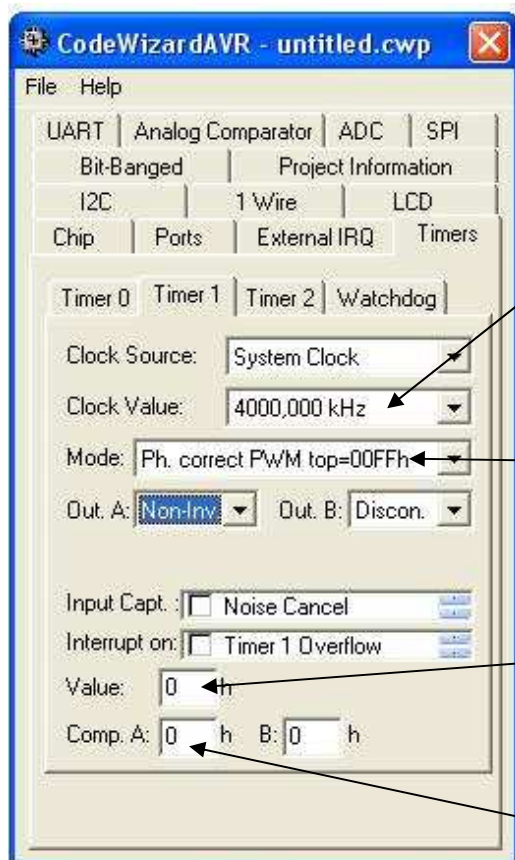
Sélectionnez l'onglet « LCD ».

Après avoir étudié le schéma de la carte « **ATMELSSI V1** », **déterminez** sur quel port est connecté l'afficheur LCD et le nombre de caractères par ligne de cet afficheur. Configurez les champs « **LCD Port** » et « **Chars./Line** » de la boîte de dialogue « LCD ».



- Configuration du timer

- Sélectionnez l'onglet « Timers » puis « Timer 1 ».



Configurez la boîte de dialogue comme ci-contre :

Fréquence  $f_{clk}$  du signal H

- Enregistrement du projet



$M_{max}$

$M_{initial}$

$N_{initial}$

- Sélectionnez « **Program Preview** ».

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

```
#include <90s8535.h>

// Alphanumeric LCD Module functions
#asm
.equ __lcd_port=0x15
#endasm
#include <lcd.h>
```

```
// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=Out Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=0 State6=T State7=T
PORTD=0x00;
DDRD=0x20;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 4000,000 kHz
// Mode: Ph. correct PWM top=0xFFh
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x81;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// LCD module initialization
lcd_init(16);
```

- Fermez la fenêtre « **Program Preview** ».
- Sélectionnez « **Generate, save and Exit** ».
- Donnez le nom **PWM** à votre projet. (sa création nécessite de définir trois fichiers)

**ATTENTION** : Le Magicien ne peut plus être utilisé pour modifier votre projet. Voir le prof pour d'éventuelles corrections.

### B3/ Etape 3 : 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.

Malgré la complexité des structures à mettre en œuvre, le programme à réaliser reste relativement simple. Ceci est dû à la « richesse » des **bibliothèques de fonctions** fournies avec le cross-compileur CodeVisionAVR.

L'écriture sur le LCD nécessite la fonction `sprintf()`. Cette fonction est située dans la librairie `stdio` accessible par le fichier `stdio.h`

#### (1) Déclaration des bibliothèques de fonctions utilisées dans le programme

Vous devez rajouter `#include <ssi.h>` (contient les définitions des noms des BP et la fonction `affiche()`), `#include <stdio.h>` et `#include <delay.h>` à la suite de `#include <90s8535.h>` dans le fichier source.

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

Le programme à réaliser peut être résumé par les actions ci-dessous :

Lire (consigne de rapport cyclique)  
Traitement (Afficher la consigne de rapport cyclique sur un LCD, sélectionner dans une table la valeur à placer dans OCR1A)  
Ecrire (la valeur choisie dans le registre OCR1A)

##### o Acquisition de la consigne de rapport cyclique (encadré ci-dessous)

La valeur du rapport cyclique (variable `rcycl`) est modifiée avec les boutons poussoir INC et DEC de la carte SSI. A chaque action sur INC, on incrémente la variable `rcycl` avec la valeur 10. A chaque Action sur Dec, on la décrémente avec la valeur 10. Un test est effectué pour que `rcycl` soit toujours compris entre 0 et 100%

Complétez le fichier source C comme ci-dessous.

```
while (1)
    { // début while

// -----Lecture de la consigne de rapport cyclique-----
    BP = Lire_BP(); // Incrémentation ou décrémentation du rapport cyclique

    switch(BP)
    {
        case INC: if (rcycl < 90) rcycl = rcycl+10; else rcycl = 100; break;
        case DEC: if (rcycl > 10) rcycl = rcycl-10; else rcycl = 0; break;
    }
}
```

##### o Traitement (sélectionner dans une table la valeur à placer dans OCR1A)

Complétez le fichier source C comme ci-dessous.

```
// ----- Traitement et modification du rapport cyclique -----

OCR1A = alpha[rcycl/10]; // Modification du rapport cyclique

if (rcycl_1 != rcycl) // On met à jour l'affichage si la valeur du rapport
                    // cyclique a été modifiée
{
    sprintf(display_buffer_ligne0, "Rap. cycl = %-u%%", rcycl);
    sprintf(display_buffer_ligne1, "INC>+ DEC>-      ");
    Affiche_LCD(display_buffer_ligne0, display_buffer_ligne1);
    rcycl_1 = rcycl;
}
```



### (3) Partie déclarative du programme à réaliser

Le programme ci-dessus utilise deux types de variables :

- alpha : tableau d'octets non signés,
- rcycl, rcycl\_1, BP : octets non signés
- display\_buffer, Chaine\_Humidite : tableaux de caractères

Pour être reconnues, ces variables doivent être **déclarées** avant leur utilisation.

Complétez le fichier source C comme ci-dessous. Remplacez ??? par vos dix valeurs séparées par une virgule:

```
void main(void)
{
// Declare your local variables here
// -----
//type          nom          désignation
// -----
char display_buffer_ligne0[17];          // tampon ligne 0 de l'afficheur
char display_buffer_ligne1[17];          // tampon ligne 1 de l'afficheur
// Coefficient pour le réglage du rapport cyclique du signal PWM
unsigned char alpha[11] = {0, ??? }; // à compléter avec vos dix valeurs
unsigned char rcycl=0, rcycl_1=0;        // rapport cyclique
unsigned char BP;                        // Permet d'identifier le bouton poussoir
                                         // actionné
}
```

## C) Programmation du composant

**Configurez le projet**

- > Project
- > Configure
  - > Sélectionnez l'onglet "After Make"
  - > Cochez "Program the Chip"

**Programmez le composant**

- > Icône "Make the Project"
- > "Program"

## D) Test du programme

### D1) Test sans la carte interface de puissance

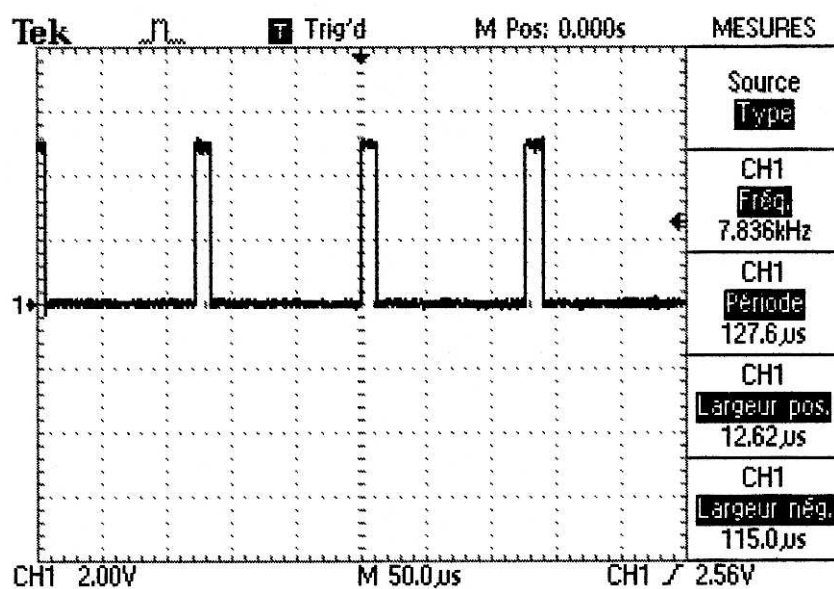
Branchez un oscilloscope entre la douille PWMA de la carte SSI et le 0V.  
Vous devez obtenir un signal TOR dont le rapport cyclique varie par pas de 10%. Voir les exemples en annexe 1.

### D2) Test avec la carte interface de puissance

*Appel prof*

Pour connexions carte de puissance et moteur.

Rapport cyclique = 10%



Rapport cyclyque = 60%

