

Fiche guide 8	TS SI		P.P.E Mini serre	 académie d'Orléans-Tours É Éducation nationale enseignement supérieur recherche 
Analyse	2h			
	<b>Commande d'un moteur à courant continu avec un signal PWM</b>			

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

**Objectif** : Contrôler la fréquence de rotation du moteur du ventilateur de la serre.

**Matériels**  
Carte ATMELSSI V1 + Module interface de puissance + Alimentation 10V.

**Logiciels**  
CodeVisionAvr V2.

**Documentation**  
Schémas de la carte ATMELSSI V1 et de la carte interface de puissance. Documentation technique de l'afficheur LCD à processeur Hitachi. Résumé de langage C.

**Sur le site WebGE** à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)  
Le présent document et la documentation technique des composants sont téléchargeables

**Sommaire**

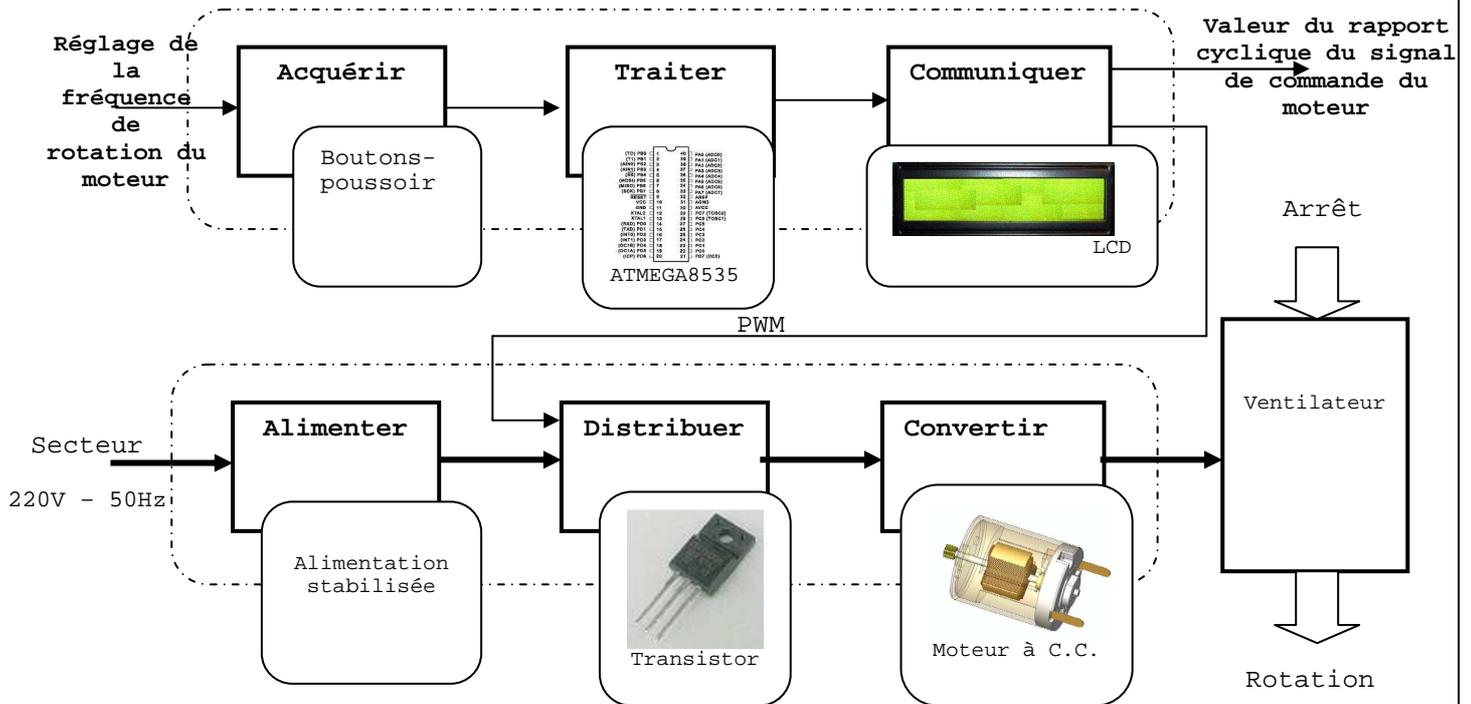
- A) Mise en situation
  - A1) Schéma fonctionnel
  - A2) Principe de réglage de la fréquence de rotation d'un moteur à courant continu
- B) Travail demandé
  - B1) Etape 1 : Détermination de  $OCR1A=f(\alpha)$
  - B2) Etape 2 : Création et configuration d'un projet CVAVR
  - B3) Etape 3 : Ecriture du programme
- C) Programmation du composant
- D) Test du programme

Corrigé le	Commentaires

## A) Mise en situation

### A1) Schéma fonctionnel

On souhaite faire varier la fréquence de rotation du moteur de ventilateur de la serre. Pour cela, on 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 entrainer 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 consiste à **réaliser le logiciel** à implanter dans le microcontrôleur.

Pour cela, vous allez **créer et configurer un projet** avec le magicien du cross-compilateur **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.<sup>1</sup> présente à ses bornes en le commandant par un signal dit **Modulé en Largeur d'Impulsion « MLI (ou PWM) »**<sup>2</sup>.

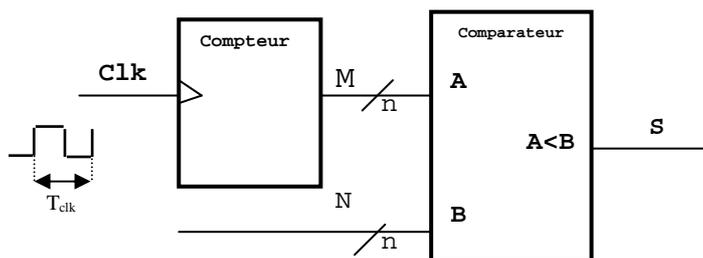
- Génération d'un signal Modulé en Largeur d'Impulsion (principe)

Un signal modulé en largeur d'impulsion peut être obtenu à partir d'un signal périodique **Clk** 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  **$T = 2^n \cdot T_{clk}$**  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  $S_{maxi}$ .

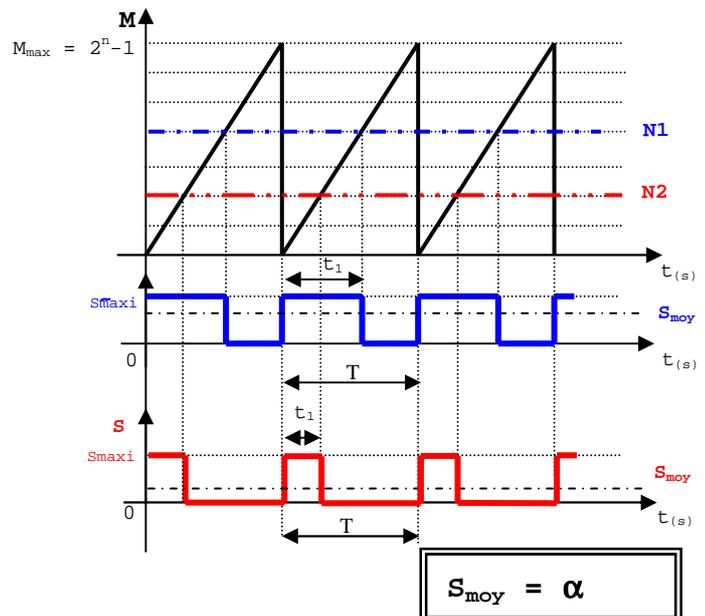
On donne ci-dessous le schéma de principe d'une structure générant un signal M.L.I et les chronogrammes de **S(t)** pour deux valeurs particulières de **N**.



**M** = valeur numérique variable ( $0 \leq M \leq 2^n - 1$ )  
**N** = valeur numérique constante

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

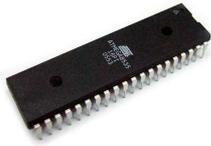
**Clk** : signal périodique de fréquence  $F_{clk}$



<sup>1</sup> d.d.p : différence de potentiel

<sup>2</sup> M.L.I : Modulation de largeur d'impulsion (P.W.M. : Pulse With Modulation)

• Génération d'un signal MLI avec le microcontrôleur ATMEGA8535

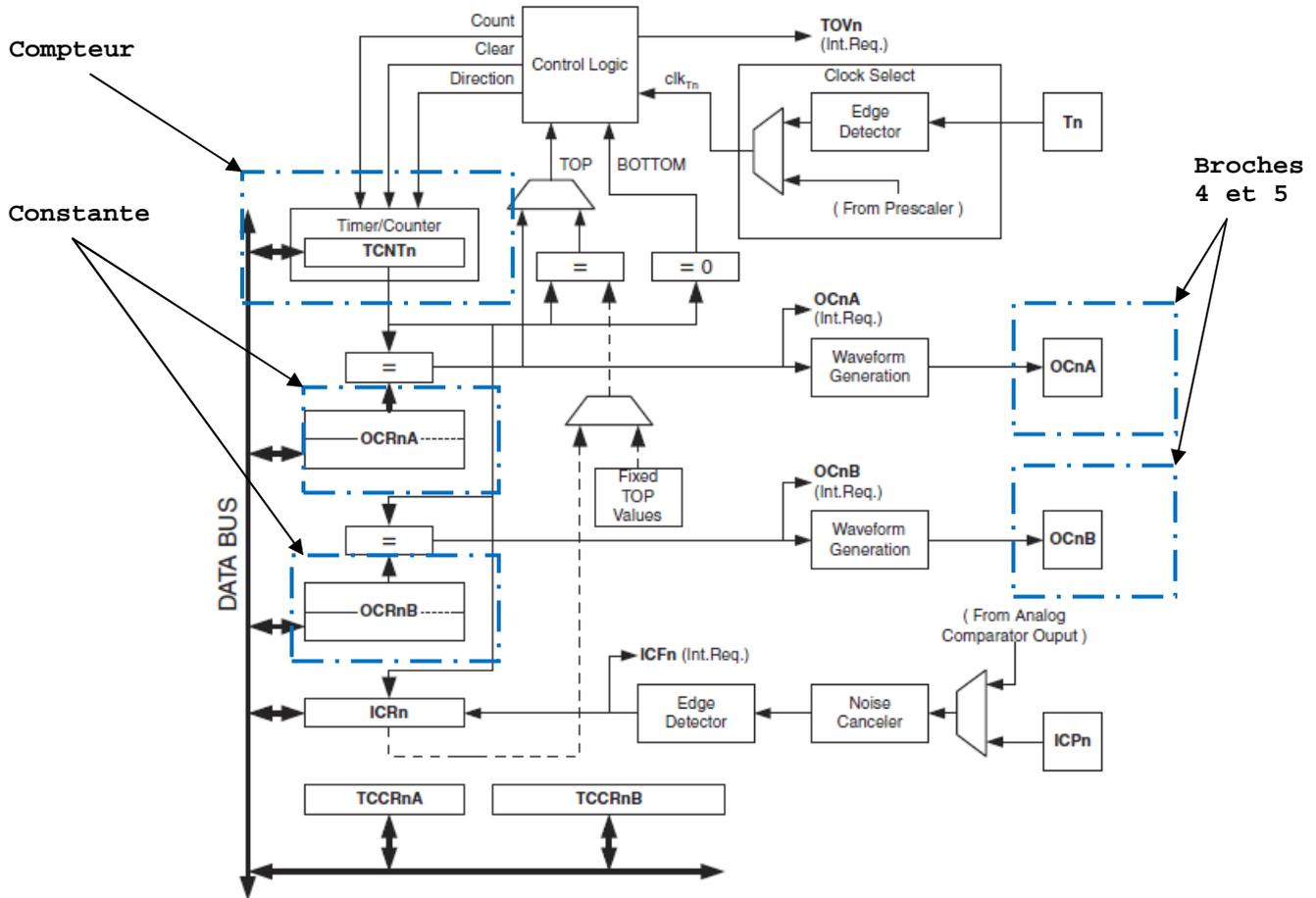


Le **Timer 1** de l'ATMEGA8535 permet de générer deux signaux modulés en largeur d'impulsion. Ces signaux sont identifiés OC1B et OC1A. Ils sont accessibles sur les broches 4 et 5 du microcontrôleur (port D).

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'ATMEGA8535



Le compteur TCNTn ( $n=1$  ici) génère le signal numérique M. Les registres OCRnA et OCRnB correspondent à la valeur numérique constante N. Les broches OCnA et OCnB correspondent à la sortie S du comparateur.

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. Pour régler une valeur particulière de  $\alpha$ , il suffit de connaître la relation existant entre  $\alpha$  et OCR1A.

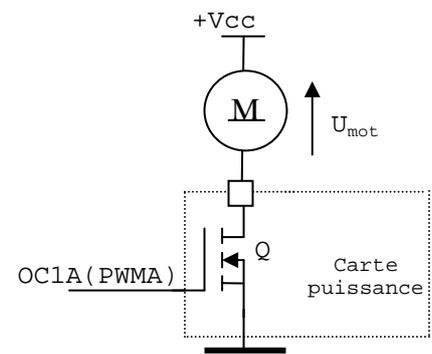
Vous allez établir cette relation dans la suite de ce document.



- 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 entrainer le moteur à courant continu du ventilateur, la fonction « Distribuer » (carte de puissance représentée ci-contre) se charge de l'adapter.

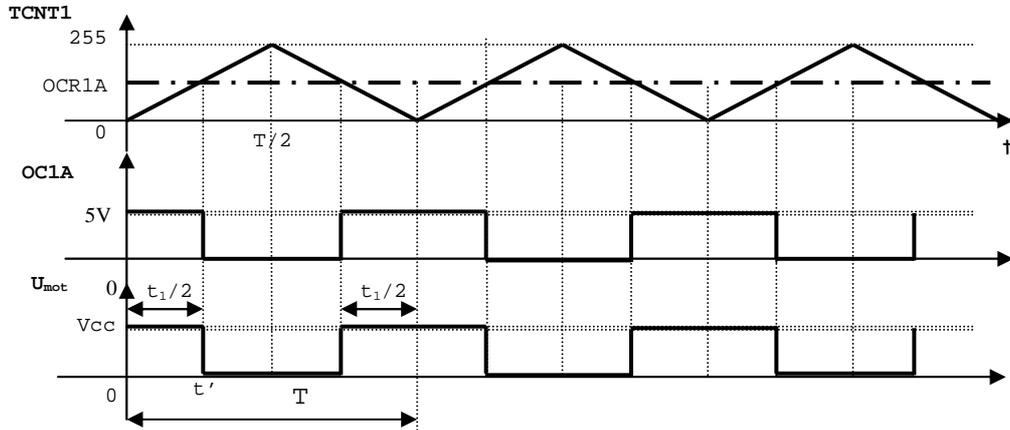


## B) Travail demandé

### B1) Etape 1 : Détermination de $OCR1A=f(\alpha)$

**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 :



Q1) Exprimez  $TCNT1 = f(t)$  pour  $t \in [0, T/2]$

---

Q2) Exprimez  $t' = f(t_1)$  (1)

---

Q3) A  $t = t'$ ,  $TCNT1 = OCR1A$ , exprimez  $t' = f(OCR1A)$  (2)

---



---



---

Q4) Exprimez  $OCR1A = f(\alpha)$  à partir des expressions (1) et (2)

---



---



---

Q5) 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 CVAVR

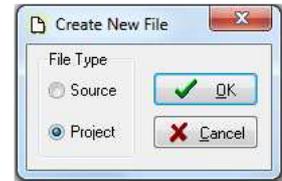
Démarrez le logiciel CodeVisionAVR



### (1) Création d'un nouveau projet

Dans la barre d'outils : « File » puis « New » pour obtenir la boîte de dialogue ci-contre.

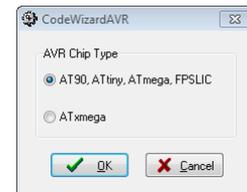
Cochez « Project » puis clic sur « Ok »



Ici « Yes »

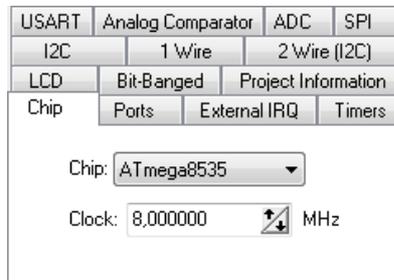


puis « OK »



### (2) Sélection du composant cible

La boîte du « Magicien » ci-dessous s'ouvre. Choisissez le « Chip » ATMEGA8535 et réglez le signal d'horloge « Clock » à 8Mhz.



### (3) 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 »)

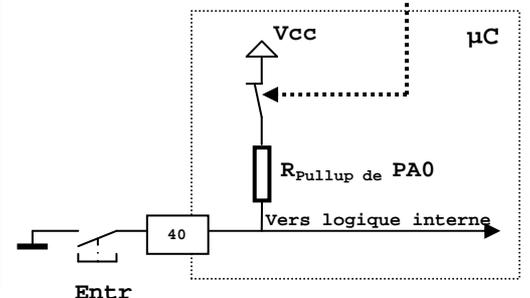
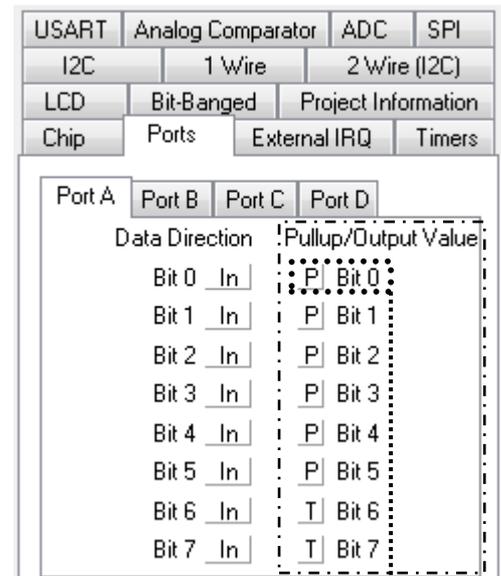
- Configurez le Port A comme ci-contre. Sélectionnez les résistances de <Pullup>.

#### Explications concernant la résistance de Pullup

Chaque broche reliée à un port d'entrée sortie du microcontrôleur est dotée d'une résistance dite de « Pullup ». Cette résistance est susceptible d'être reliée, par le logiciel, au potentiel positif de l'alimentation. Elle permet de **fixer** la valeur du potentiel sur la broche concernée.

Exemple : Sur la carte SSI, le BP <Entr> est relié à la broche 40 (PA0) du microcontrôleur. Lorsque la résistance de Pullup est connectée, cette broche « voit » un niveau logique « 1 » si <Entr> est ouvert et un niveau logique « 0 » si il est fermé. Sans cette résistance, <Entr> ouvert produirait un état logique « aléatoire » (« 0 » ou « 1 ») ;

- Sélectionnez l'onglet « Port D ».



**Q6) Configurez le bit 5 du port D après avoir déterminé son sens (entrée ou sortie).**

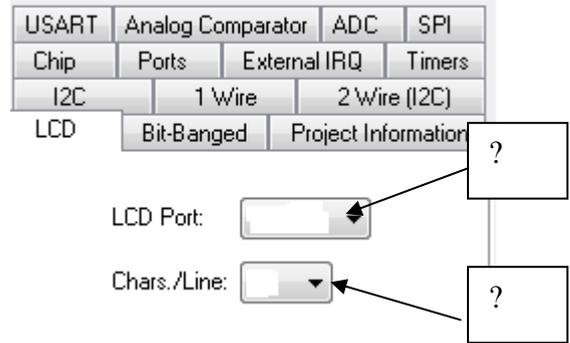


**(4) Choix de l'affichage**

Sélectionnez l'onglet « LCD ».

Pour obtenir l'organisation ci-contre, il est nécessaire de remplir le champ « LCD Port ».

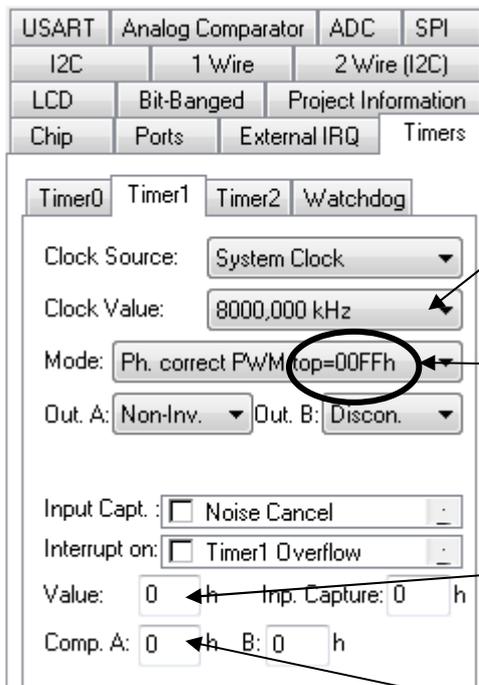
En étudiant le schéma de la carte « ATMELSSI V1 », et la documentation de l'afficheur LCD, **déterminez** sur quel port est connecté l'afficheur LCD et le nombre de caractères par ligne que comporte cet afficheur.



**Q7) Configurez** les champs « LCD Port » et « Chars./Line » de la boîte de dialogue « LCD ».

**(5) Configuration du timer 1**

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



- Configurez la boîte de dialogue comme ci-contre :

Fréquence  $f_{clk}$  du signal H (§A2)

$M_{max}$  (§A2)

$M_{initial}$

$N_{initial}$

**(6) 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 <mega8535.h>

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

// Declare your global variables here

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

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=P State4=P State3=P State2=P State1=P State0=P
PORTA=0x3F;
DDRA=0x00;

                                     Plus loin dans le fichier source

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

                                     Plus loin dans le fichier source

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 8000,000 kHz
// Mode: Ph. correct PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x81;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

                                     Plus loin dans le fichier source

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

**Fermez** la fenêtre.

**Sélectionnez**

→ File

→ « **Generate, save and Exit** ».

**Donnez** le nom **PWM** à votre projet (**3 fois**) pour créer les trois fichiers de base du projet. (.c, .prj, .cwp)

 **ATTENTION** : Après la sauvegarde du projet, le magicien ne peut plus être utilisé pour le modifier. Voir le prof pour d'éventuelles corrections.

### B3) Etape 3 : Ecriture du programme

Dans ce paragraphe, 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 `printf()`. Cette fonction est située dans la bibliothèque `stdio` accessible par une référence au fichier `stdio.h`

Une référence à une bibliothèque se fait à l'aide de la **directive de compilation** **`#include <nom bibliothèque>`**.

Exemple : **`#include <stdio.h>`**

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

Vous **devez rajouter** une référence à la bibliothèque **`ssi2`** (définitions des noms des BP et la fonction `affiche()`), à la bibliothèque **`stdio`** et à la bibliothèque **`delay`** (nécessaire pour réaliser une temporisation) à la suite de `#include <mega8535.h>` dans le fichier source.

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

Le programme à réaliser doit contenir les actions ci-dessous:

**Lire** (la consigne de rapport cyclique)  
**Traiter** (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)

- **Code de « Lire la consigne de rapport cyclique » (voir 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;
    }
}
```

Analyse du code "Lire la consigne de rapport cyclique"

Q8) Dessinez l'algorithme correspondant au code C ci-dessus.

Début

- o Code de « Traiter... et Ecrire... »

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;
}
```





Q12) Dans l'expression  $OCR1A = \alpha[rcycl/10]$  ; pourquoi  $rcycl$  est il divisé par 10 ?

## C) Programmation du composant

Configurez le projet

-> Project

-> Configure

-> Sélectionnez l'onglet "After Build"

-> Cochez "Program the Chip"

Compilez le projet par l'icône "Build the Project"



Téléchargez le programme dans la carte (Bouton « 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% lorsque vous appuyez sur les boutons-poussoirs <Inc> et <Dec>.

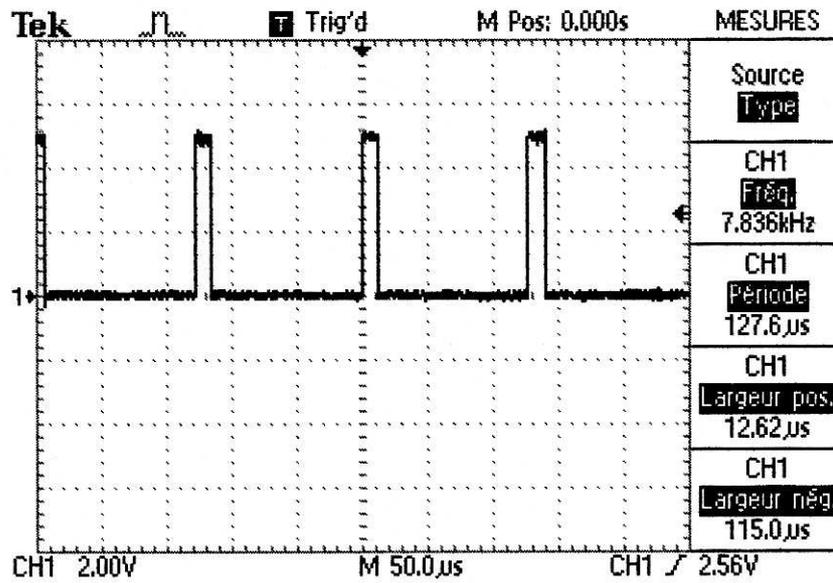
Voir les exemples en annexe 1.

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

*Appel prof*

Pour connexions de la carte de puissance et du moteur.

Rapport cyclique = 10%



Rapport cyclique = 60%

