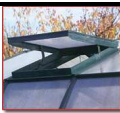




Fiche guide 7	TS SI		P.P.E. Mini serre	
	2h			
 Lycée Polyvalent PIERRE EMILE MARTIN	Commande d'un moteur à courant continu par un signal PWM			

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

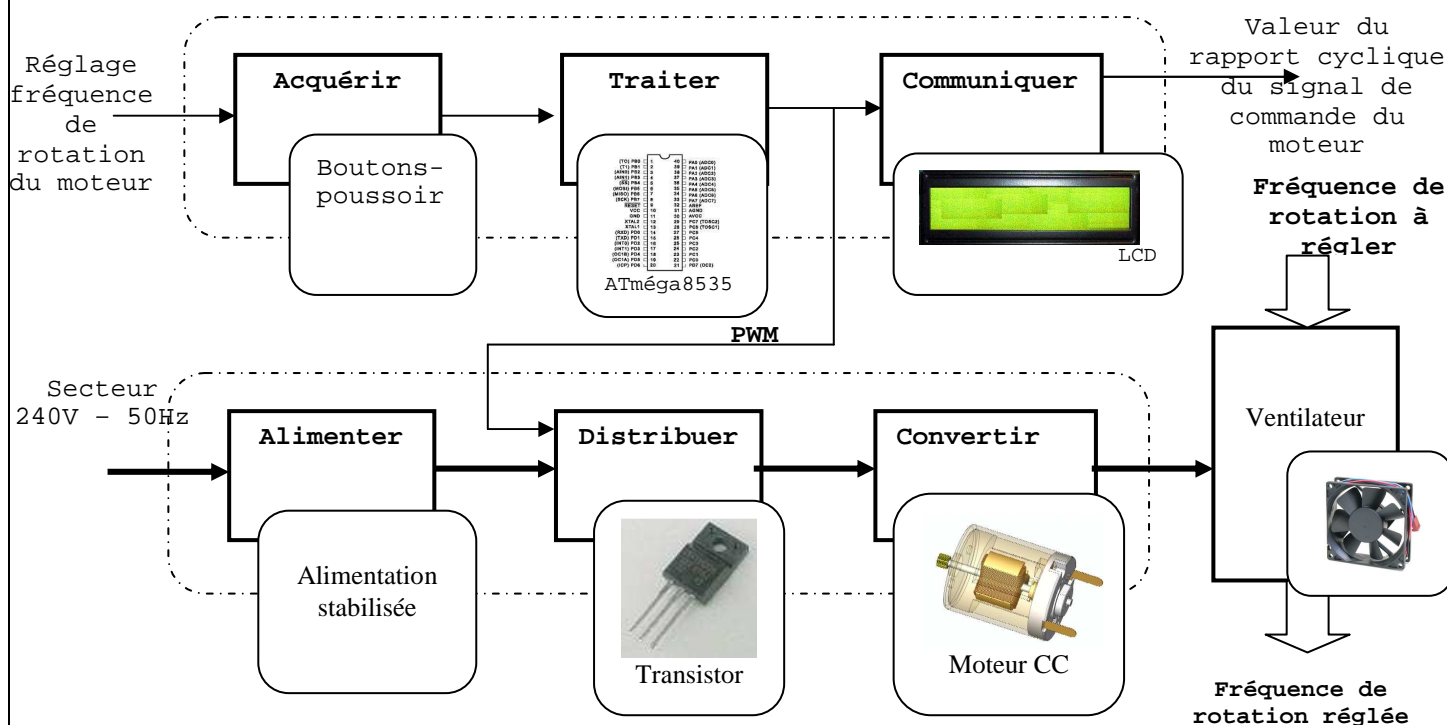
**Objectif :** Contrôler la fréquence de rotation du moteur du ventilateur de la serre.



<b>Matériels</b> Carte ATMELSSI V1 + Module interface de puissance + Alimentation 10V.
<b>Logiciels</b> CodeVisionAvr.
<b>Documentation</b> Schémas de la carte ATMELSSI V1 et de la carte interface de puissance. Documentation technique de l'afficheur LCD à processeur Hitachi.
Le présent document et la documentation technique des composants sont <u>téléchargeables</u> sur le site <u>WebGE</u> à l'adresse <a href="http://p.mariano.free.fr/">http://p.mariano.free.fr/</a> (rubrique PPE).

## A) Présentation

### A1) Schéma fonctionnel

On souhaite **contrôler** 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.



 Lycée Polyvalent PIERRE EMILE MARTIN	FG7	PWM		PPE Mini Serre	1
--	-----	-----	---	----------------	---

La fonction « Traiter » est assurée par un programme implanté dans le microcontrôleur ATMega8535. Le signal PWM est issu d'une structure appelée « Timer ». Celle-ci est intégrée au microcontrôleur. 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 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)\*** ».

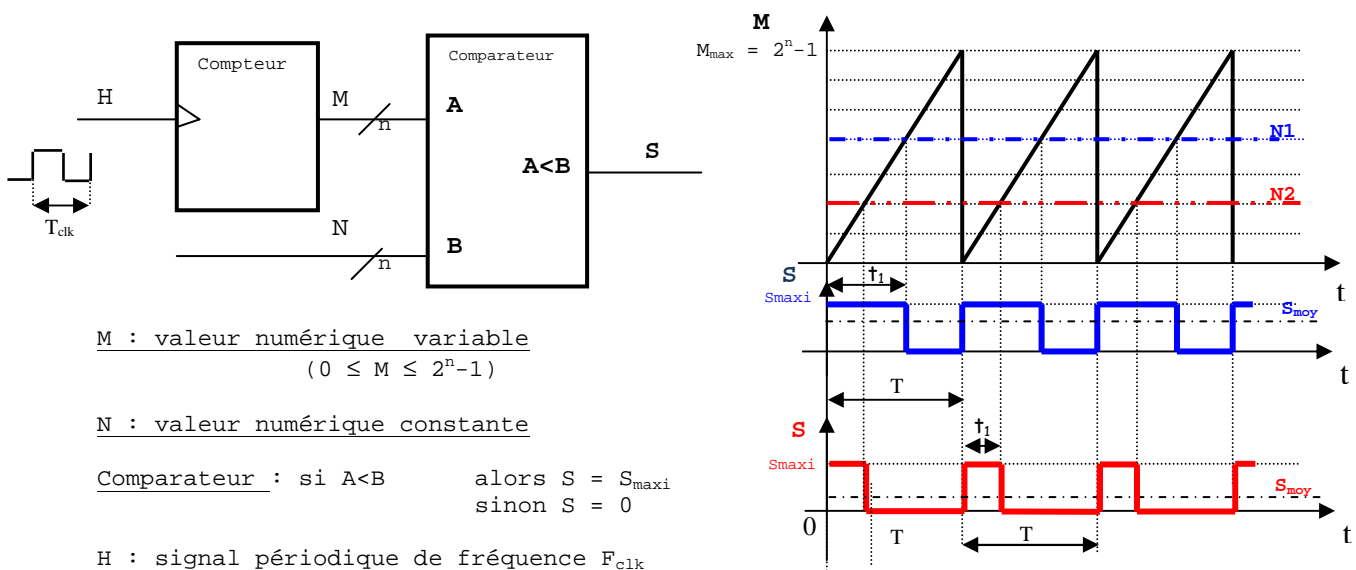
- Génération d'un signal Modulé en Largeur d'Impulsion (principe)

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 (N1 et N2 ci-dessous).

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.

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

\*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 des 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)

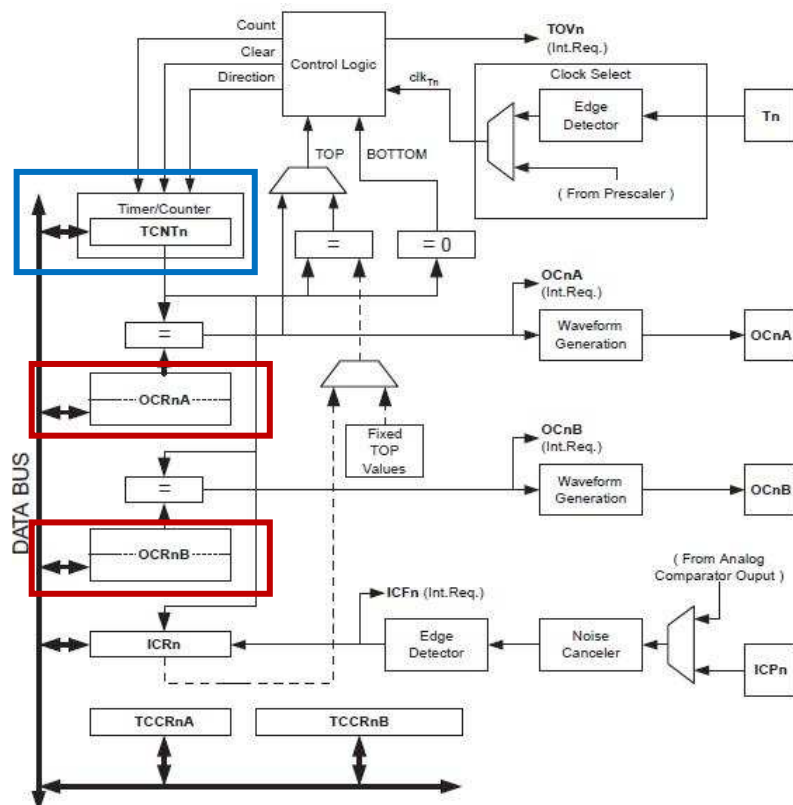
Le **Timer 1** intègre un **compteur**, des **comparateurs** et divers registres. En mode M.L.I. son fonctionnement répond au principe exposé dans le paragraphe précédent.

5

PDIP

(XCK/T0) PB0	1	40	PA0 (ADC0)
(T1) PB1	2	39	PA1 (ADC1)
(INT2/AIN0) PB2	3	38	PA2 (ADC2)
(OC0/AIN1) PB3	4	37	PA3 (ADC3)
(SS) PB4	5	36	PA4 (ADC4)
(MOSI) PB5	6	35	PA5 (ADC5)
(MISO) PB6	7	34	PA6 (ADC6)
(SCK) PB7	8	33	PA7 (ADC7)
RESET	9	32	AREF
VCC	10	31	GND
GND	11	30	AVCC
XTAL2	12	29	PC7 (TOSC2)
XTAL1	13	28	PC6 (TOSC1)
(RXD) PD0	14	27	PC5
(TXD) PD1	15	26	PC4
(INT0) PD2	16	25	PC3
(INT1) PD3	17	24	PC2
(OC1B) PD4	18	23	PC1 (SDA)
(OC1A) PD5	19	22	PC0 (SCL)
(ICF1) PD6	20	21	PD7 (OC2)

Schéma fonctionnel du timer 1 de l'ATmega8535



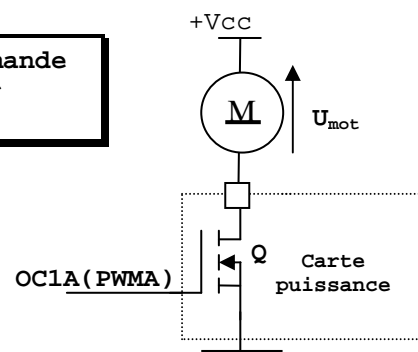
Le compteur TCNT1 génère le signal numérique M. Les registres OCR1A et OCR1B correspondent à N. Les broches OC1A et OC1B correspondent à S. Le Timer 1 contient donc deux structures dont le fonctionnement répond au principe exposé dans le paragraphe précédent.

La modification du rapport cyclique  $\alpha$  du signal de commande du moteur du ventilateur se fera 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 entrainer le moteur à courant continu du ventilateur, la fonction « Distribuer » (carte de puissance représentée ci-dessus) se charge de l'adapter.

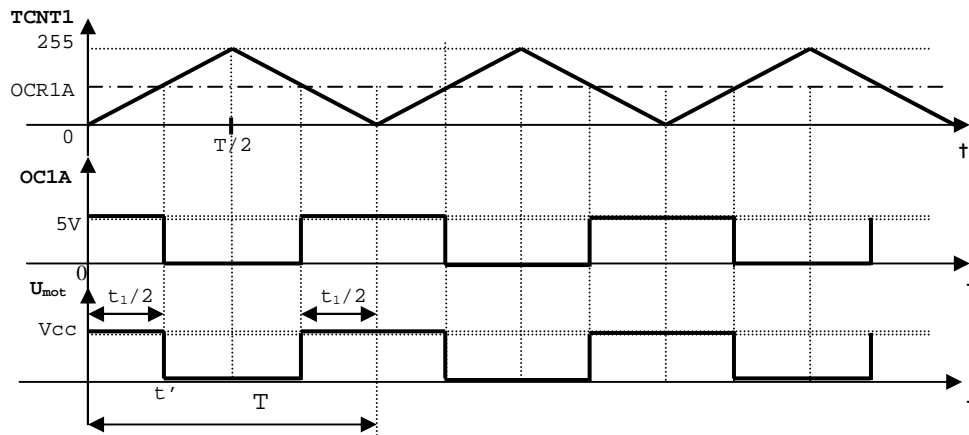


## B) Travail demandé

### B1/ Etape 1 : Détermination des valeurs à placer dans le registre OCR1A pour régler la valeur du rapport cyclique $\alpha$

**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.

Les chronogrammes ci-dessous sont extraits des « datasheets » ATMEL. Pour établir  $\alpha = f(\text{OCR1A})$ , il faut « raisonner » avec le signal  $U_{\text{mot}}$  présent aux bornes du moteur du ventilateur. Il faut donc tenir compte du transistor Q.



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

---

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

---

Q1c) A  $t = t'$ ,  $\text{TCNT1} = \text{OCR1A}$ , exprimez  $t' = f(\text{OCR1A})$  (2)

---



---



---

Q1d) Exprimez  $\text{OCR1A} = f(\alpha)$  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
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** »

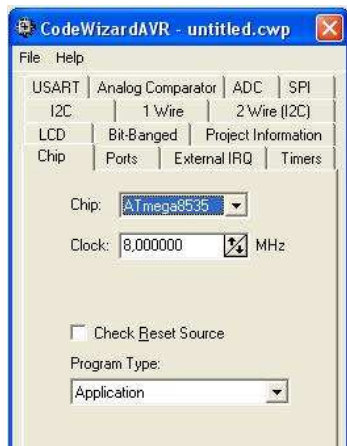


Ici « **Yes** »



### • Sélection du composant cible

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

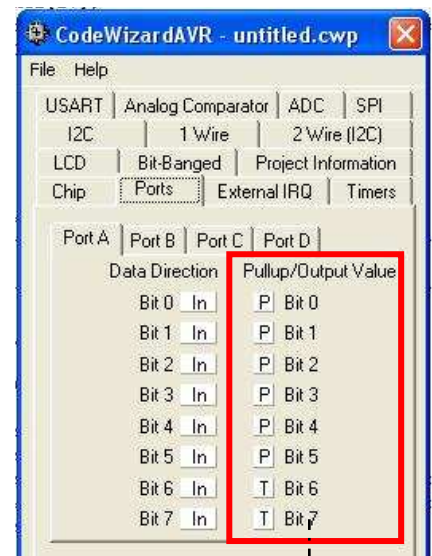


### • Configuration des ports A et D

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

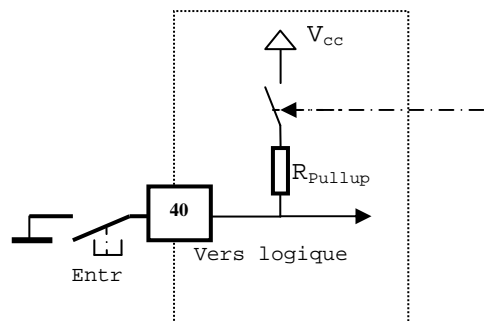
Les boutons-poussoirs de la carte ATMEL SSI sont connectés au Port A du microcontrôleur. (voir schéma structurel « **ATMELSSI V1** »).

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



**Explication concernant les résistances de Pullup**  
Chaque broche reliée à un port d'entrées sorties du microcontrôleur est dotée d'une résistance dite de « **Pullup** ». Celle-ci est susceptible d'être reliée, par le logiciel, au potentiel positif de l'alimentation. Cette résistance permet de **fixer** le potentiel sur la broche concernée.

**Exemple** : Sur la carte SSI, le BP Entr est relié à la broche 40 du microcontrôleur. Lorsque la résistance de Pullup est connectée, cette broche « voie » un niveau logique 0 si Entr est fermé et un niveau logique 1 si il est ouvert. En l'absence de cette résistance le niveau logique serait **indéterminé** lorsque Entr est ouvert.



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

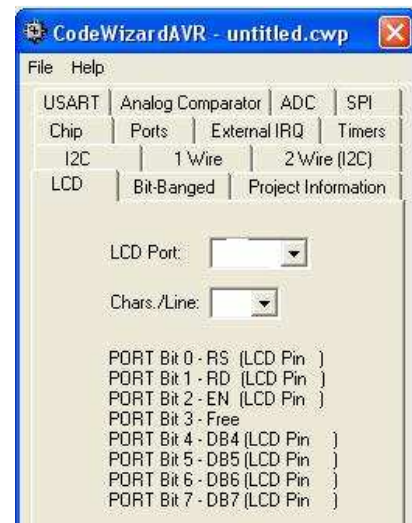
Configurez le bit 5 du port D après avoir déterminé son sens (entrée ou sortie).

## Choix de l'affichage

- Sélectionnez l'onglet « LCD ».

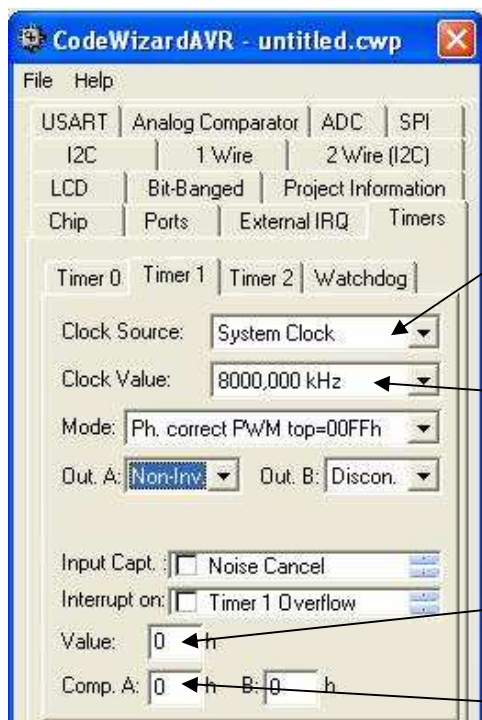
Après avoir étudié le schéma de la carte « **ATMELSSI V1** », **déterminez** sur quel port est connecté l'afficheur LCD et le nombre de caractères par ligne de cet afficheur.

**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



- 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
#endasm
#include <lcd.h>
```



```

// Declare your global variables here

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

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

.....plus loin

// 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;

.....plus loin

// 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
TCCR1A=0x81;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

.....plus loin

// LCD module initialization
lcd_init(16);

```

**Fermez** la fenêtre.

**Sélectionnez**

→ File

→ **Generate, save and Exit**

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

**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 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`. On y accède par une **référence** au fichier `<stdio.h>`

On fait référence à une bibliothèque avec la directive **#include**.

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

Vous devez rajouter une référence pour la bibliothèque `ssi` (nécessaire pour accéder aux fonctions `Lire_BP` et `Affiche_LCD`) et pour la bibliothèque `delay` (nécessaire pour réaliser une temporisation) à la suite de `#include <mega8535.h>`.

#### (2) **Partie déclarative du programme à réaliser**

Le programme ci-dessus utilise deux types de variables :

- **alpha** : `alpha` est un tableau d'octets non signés, il contient les valeurs de réglage du rapport cyclique déterminées dans l'étape 1
- **rcycl, rcycl\_1, BP** : octets non signés
- **display\_buffer\_ligne0, display\_buffer\_ligne1** : 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=1;        // rapport cyclique
unsigned char BP;                        // Permet d'identifier le bouton-poussoir
                                           // actionné
```



Le tableau alpha[11] peut être représenté comme ci-dessous :

i	alpha(i)
0	0
1	26
...	...
10	255

**Lisez** le paragraphe sur les vecteurs du document « Résumé de langage C ».

On souhaite placer la valeur permettant de régler  $\alpha = 40\%$  dans le registre OCR1A.

**Ecrivez** ci-dessous l'instruction permettant d'effectuer cette opération

---

## (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 la valeur à placer dans OCR1A dans une table)  
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 du projet comme ci-dessous.

```
sprintf(display_buffer_ligne0,"Rap. cycl = %-u%",rcycl);// Initialisation de
sprintf(display_buffer_ligne1,"INC>+ DEC>-      ");          // l'affichage
Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);

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 « Lecture de la consigne »

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

Début

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

Complétez le fichier source C du projet comme ci-dessous.

```
// ----- Traitement et modification du rapport cyclique-----  
  
OCR1A = alpha[rcycl/10]; // Mofification 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;  
}
```

Synthèse : Analyse de la partie « Traitement et modification du rapport cyclique »  
 Dans la partie exécutive du programme le tableau alpha est utilisé dans l'expression suivante :

**`OCR1A = alpha[rcycl/10];`**

Pourquoi rcycl est-il divisé par 10 ?

---



---



---

Quelle est la valeur placée dans OCR1A si le rapport cyclique affiché est égale à 60% ?

---



---

### C) Programmation du composant

Configurez le projet

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

Programmez le composant



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

### D) Test du programme

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

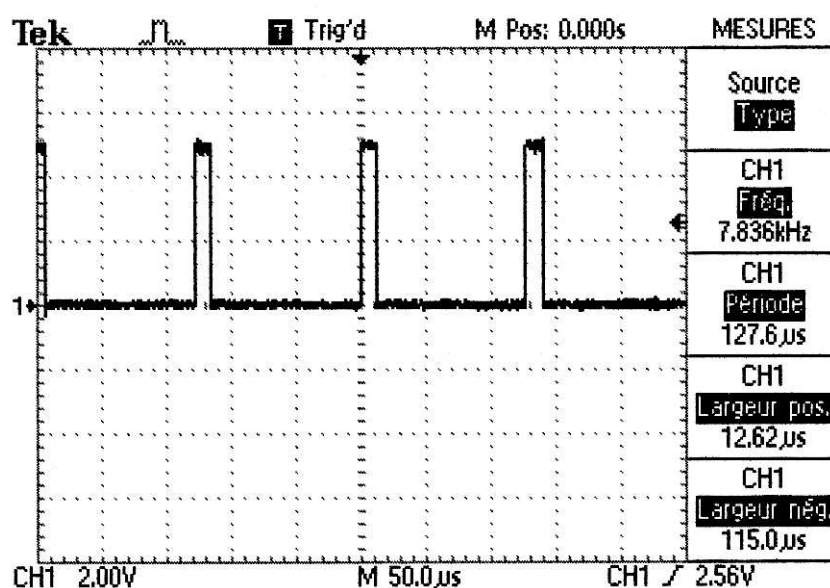
Branchez un oscilloscope entre la douille PWMA de la carte SSI et le 0V.  
 Vous devez obtenir un signal TOR dont le rapport cyclique varie par pas de 10%. Voir les exemples en annexe 1.

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

*Appel prof*

Pour connecter la carte de puissance au ventilateur.

Rapport cyclique = 10%



Rapport cyclique = 60%

