





| | | | | |
|---|--|---|--|---|
| Fiche guide 2 | TS SI |  | P.P.E. Effet de lumière et commande DMX512 |  académie d'Orléans-Tours éducation nationale enseignement supérieur recherche  France Lycée - Collège - Université Ministère de l'Éducation Nationale |
| Analyse et synthèse | | | | |
|  Lycée Polyvalent PIERRE EMILE MARTIN | Réglage de la luminosité des LED du projecteur | | | |

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

Objectif

Réaliser le programme de réglage de la luminosité des LED du projecteur à réaliser.

Matériels : 1 carte SSI + 1 carte étude LED haute luminosité.

Logiciels : CodeVisionAVR.

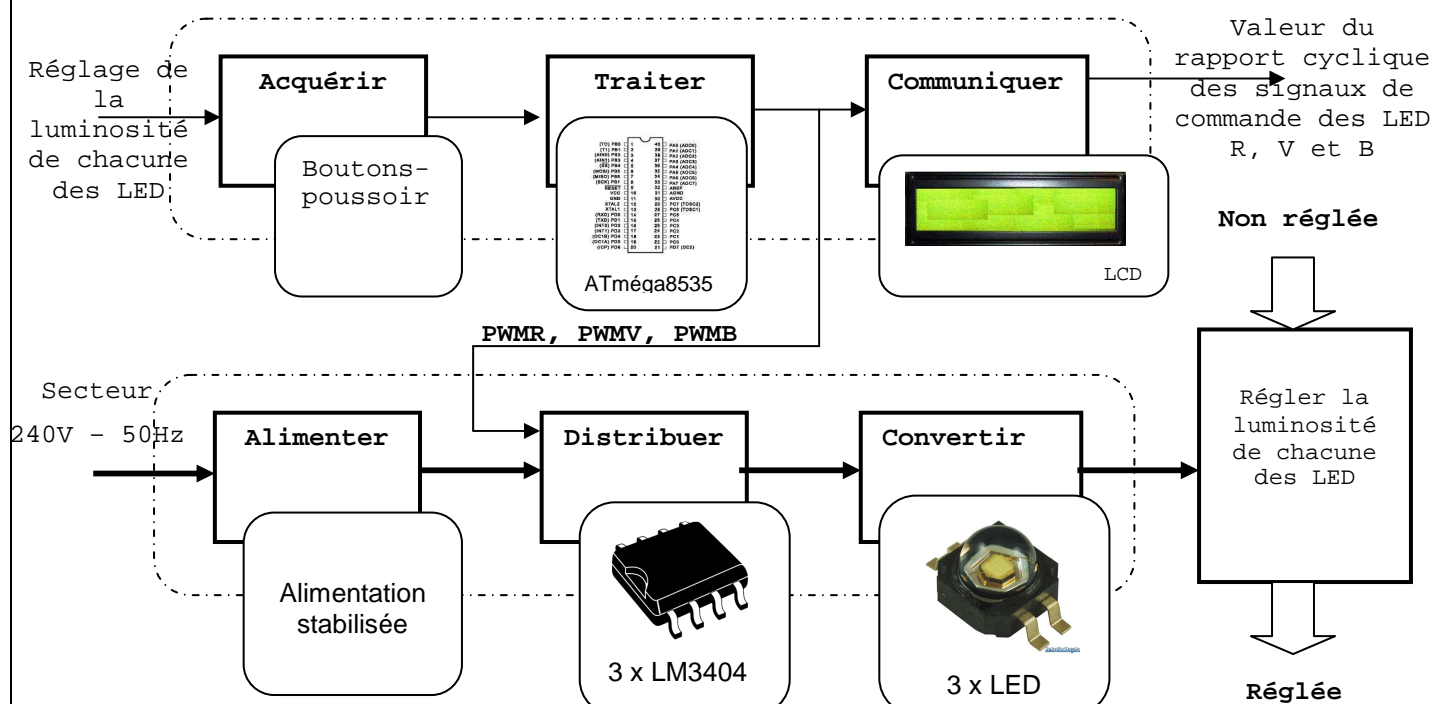
Documentation : Schéma carte SSI + Schéma carte étude LED haute luminosité. Résumé de langage C.

Le présent document est téléchargeable sur le site WebGE à l'adresse <http://p.mariano.free.fr/> (rubrique PPE)

A) Présentation

A1) Schéma fonctionnel

On souhaite **contrôler la luminosité des LED rouge, verte et bleu du projecteur à réaliser**. Pour cela, on propose de mettre en œuvre une structure correspondant au schéma ci-dessous.



| | | | | | |
|---|-----|----------------------------|---|-----------------------|---|
|  Lycée Polyvalent PIERRE EMILE MARTIN | FG2 | Réglage luminosité des LED |  | PPE Effets de lumière | 1 |
|---|-----|----------------------------|---|-----------------------|---|

La fonction « Traiter » est assurée par un programme implanté dans le microcontrôleur AtMega8535.

Les signaux PWM (PWMR, PWMV, PWMB) sont issus de structures appelées « Timer ». Celles-ci sont intégrées au microcontrôleur. PWMR, PWMV et PWMB sont des signaux de commande. Ils permettent à la fonction « Distribuer » d'ajuster l'énergie appliquée aux LED. Cette fonction est réalisée par un circuit intégré LM3404. Elle sera étudiée prochainement. La fonction « Communiquer » est remplie par un afficheur LCD (processeur Hitachi).

L'ensemble des structures matérielles étant réunies sur les cartes « ATMELSSI » et « LED haute luminosité », 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 des LED 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 luminosité des LED du projecteur à réaliser.

A2) Réglage de la luminosité des LED

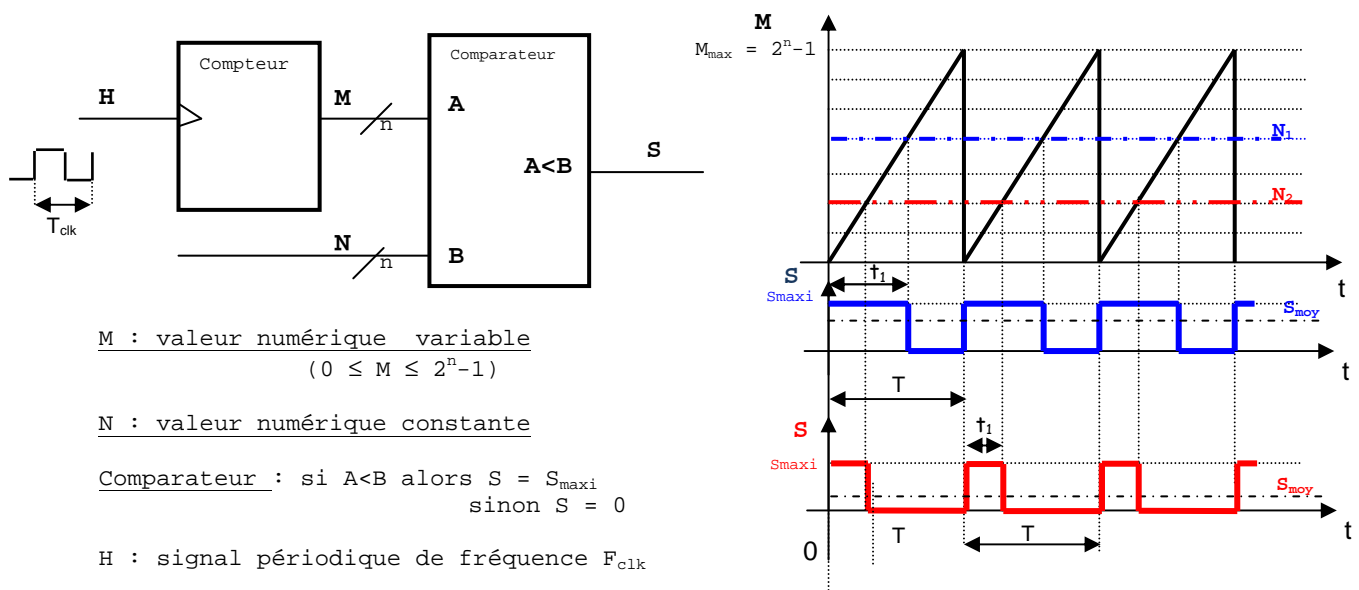
Chaque LED (ou groupe de LED de même couleur) est commandée par un **régulateur de courant** de type LM3404 spécialement conçu pour cette application. Celui-ci se charge d'ajuster l'intensité qui traverse la LED. Ce régulateur peut également régler la luminosité des LED si on applique un signal modulé en largeur d'impulsions sur une de ses entrées.

- Génération d'un signal **Modulé en Largeur d'Impulsion** : **MLI** ou **PWM**(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 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.



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 de signaux MLI avec le microcontrôleur ATmega8535**

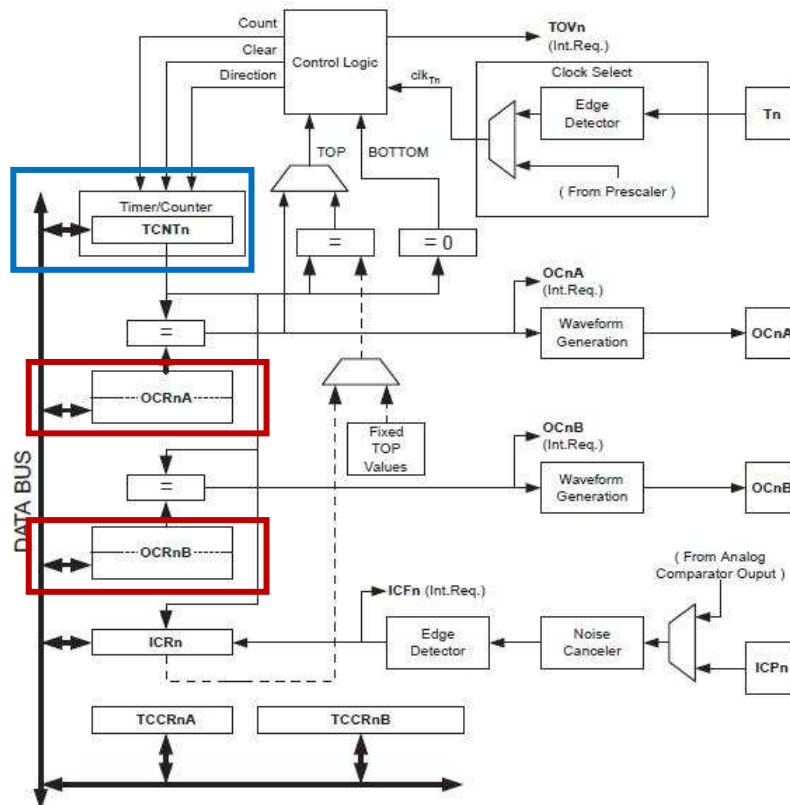


Le **Timer 1** de l'ATmega8535 permet de générer des signaux **modulés en largeur d'impulsion**. Ces signaux sont identifiés par OC1B et OC1A. Ils sont accessibles sur les broches 4 et 5 du port D. (voir ci-contre)

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.

| PDIP | |
|-----------------|-------------|
| (XCK/T0) PB0 | 1 |
| (T1) PB1 | 2 |
| (INT2/AIN0) PB2 | 3 |
| (OC0A/IN1) PB3 | 4 |
| (SS) PB4 | 5 |
| (MOSI) PB5 | 6 |
| (MISO) PB6 | 7 |
| (SCK) PB7 | 8 |
| RESET | 9 |
| VCC | 10 |
| GND | 11 |
| XTAL2 | 12 |
| XTAL1 | 13 |
| (RXD) PD0 | 14 |
| (TXD) PD1 | 15 |
| (INT0) PD2 | 16 |
| (INT1) PD3 | 17 |
| (OC1B) PD4 | 18 |
| (OC1A) PD5 | 19 |
| (ICP1) PD6 | 20 |
| 40 | PA0 (ADC0) |
| 39 | PA1 (ADC1) |
| 38 | PA2 (ADC2) |
| 37 | PA3 (ADC3) |
| 36 | PA4 (ADC4) |
| 35 | PA5 (ADC5) |
| 34 | PA6 (ADC6) |
| 33 | PA7 (ADC7) |
| 32 | AREF |
| 31 | GND |
| 30 | AVCC |
| 29 | PC7 (TOSC2) |
| 28 | PC6 (TOSC1) |
| 27 | PC5 |
| 26 | PC4 |
| 25 | PC3 |
| 24 | PC2 |
| 23 | PC1 (SDA) |
| 22 | PC0 (SCL) |
| 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.

Le Timer 0 contient également **une** structure dont le fonctionnement répond au principe exposé dans le paragraphe précédent.

La modification du rapport cyclique α des signaux de commande de la luminosité des LED rouge, verte et bleu se fait en modifiant la valeur contenue dans les registres OCR1A, OCR1B et OCR0.

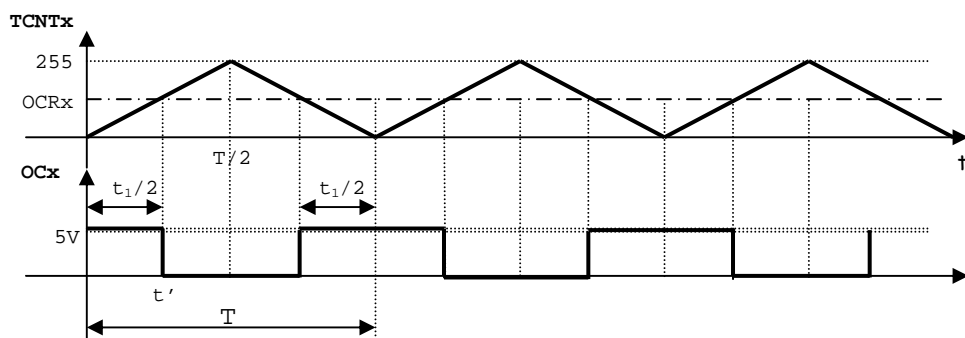


B) Travail demandé

B1/ Etape 1 : Détermination des valeurs à placer dans les registres OCRx pour régler la valeur des rapports cycliques α ($x = 1A, 1B$ ou 0)

Objectif : Déterminer les valeurs à placer dans les registres OCRx pour régler les valeurs des rapports cycliques α des signaux de commande de la luminosité des LED.

Les chronogrammes ci-dessous sont extraits des « Datasheets » ATMEL.



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

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

Q1c) A $t = t'$, $TCNTx = OCRx$, exprimez $t' = f(OCRx)$ (2)

Q1d) Etablissez $OCRx = f(\alpha)$ 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 |
|--------------|---|----|----|----|----|----|----|----|----|----|
| OCRx | | | | | | | | | | |



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.

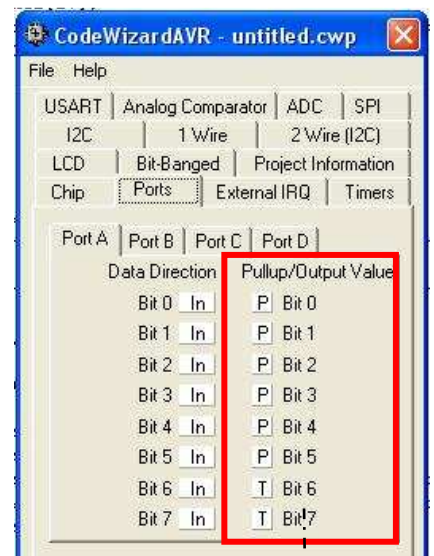
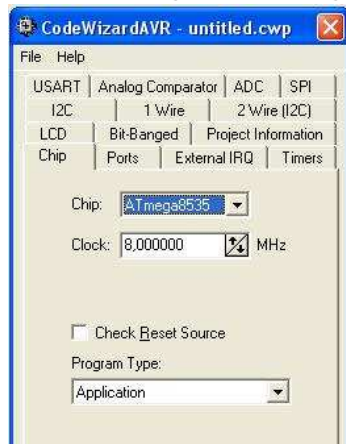
Cochez « Project » puis clic sur « Ok »



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,B et D

- Sélectionnez l'onglet « Port A ».

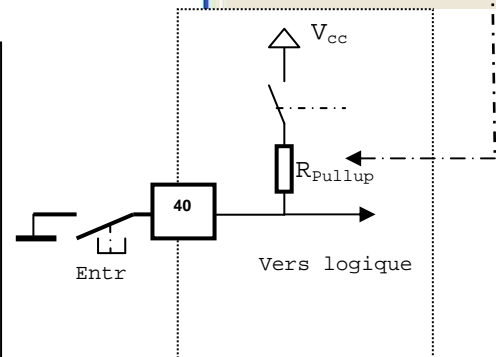
Les boutons-poussoirs de la carte ATMEAL SSI sont connectés au Port A du microcontrôleur. (voir schéma structurel « ATMEALSSI V1 »).

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 B ».

Configurez le BIT 3, du port B après avoir déterminé son sens (entrée ou sortie).



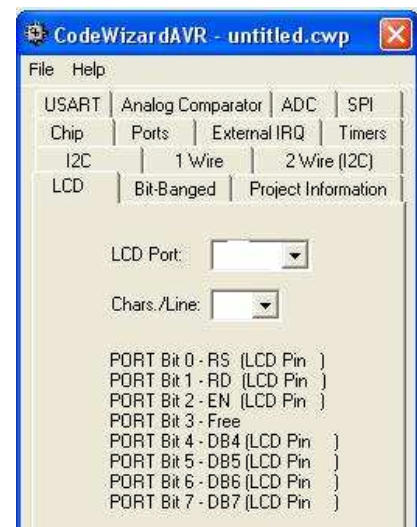
- **Sélectionnez** l'onglet « **Port D** ».
Configurez les BIT 4 et 5, du port D après avoir déterminé leur sens (entrée ou sortie).

Choix de l'affichage

- **Sélectionnez** l'onglet « **LCD** ».

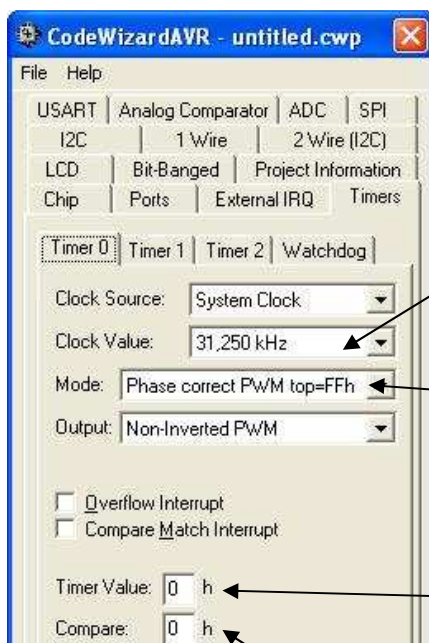
Après avoir étudié le schéma de la carte « **ATMELSSI V1** », **déterminez** sur quel port est connecté l'afficheur LCD et le nombre de caractères par ligne de cet afficheur.

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



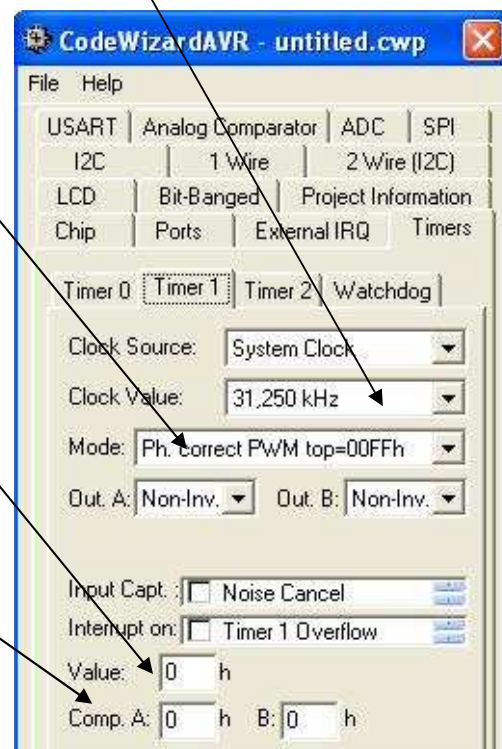
• Configuration du timer

- **sélectionnez** l'onglet « **Timers** » puis « **Timer 0** ».



Configurez la boîte de dialogue comme ci-contre :

- **sélectionnez** l'onglet « **Timer 1** ».



Fréquence f_{clk} du signal H

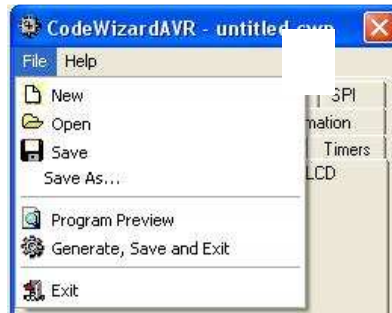
M_{max}

$M_{initial}$

$N_{initial}$



- 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;

    // Port B initialization
    // Func0=In Func1=In Func2=In Func3=Out Func4=In Func5=In Func6=In Func7=In
    // State0=T State1=T State2=T State3=0 State4=T State5=T State6=T State7=T
    PORTB=0x00;
    DDRB=0x08;

    // Port C initialization
    // Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
    // State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
    PORTC=0x00;
    DDRC=0x00;
```



```
// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=Out Func5=Out Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=0 State5=0 State6=T State7=T
PORTD=0x00;
DDRD=0x30;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 31,250 kHz
// Mode: Phase correct PWM top=FFh
// OC0 output: Non-Inverted PWM
TCCR0=0x64;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31,250 kHz
// Mode: Ph. correct PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0xA1;
TCCR1B=0x04;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

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

Fermez la fenêtre.

Sélectionnez

→ File

→ **Generate, save and Exit**

Donnez le nom **CHCOLO** 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 `sprintf()`. 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 exécutive du programme à réaliser

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

Lire (les consignes de rapport cyclique)

Traitement (Afficher les consignes de rapport cyclique sur un LCD et sélectionner, dans une table, les valeurs à placer dans les registres `OCR1A`, `OCR1B` et `OCR0`)

Ecrire (la valeur choisie dans les registres `OCR1A`, `OCR1B` et `OCR0`)



○ Acquisition des consignes de rapport cyclique (encadré ci-dessous)

Les valeurs des rapports cycliques (**variables R, V et B**) sont modifiées avec les boutons poussoir {INC, DEC} {OK, SET} {ECHAP, ENTR} de la carte SSI.

Ex : A chaque action sur INC, on incrémente la variable R avec la valeur 10.
A chaque Action sur Dec, on la décrémente avec la valeur 10.

Un test est effectué pour que R soit toujours compris entre 0 et 100%

Complétez le fichier source C comme ci-dessous.

```
sprintf(display_buffer_ligne0,"Rouge Vert Bleu");
sprintf(display_buffer_ligne1,"%-u%%  %-u%%  %-u%%",R,V,B);
Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);

while (1)
    {
        // début while
        BP = Lire_BP(); // Incrémentation ou décrémentation du rapport cyclique

        switch(BP)
        {
            case INC: if (R < 90) R = R+10; else R = 100; break;
            case DEC: if (R > 10) R = R-10; else R = 0; break;
            case OK: if (V < 90) V = V+10; else V = 100; break;
            case SET: if (V > 10) V = V-10; else V = 0; break;
            case ECHAP : if (B < 90) B = B+10; else B = 100; break;
            case ENTR : if (B > 10) B = B-10; else B = 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 comme ci-dessous.

```
// ----- Traitement et modification du rapport cyclique-----
OCR1AL=alpha[R/10]; // Modification des rapports cycliques
OCR1BL=alpha[V/10];
OCR0=alpha[B/10];

// On met à jour l'affichage si la valeur du rapport cyclique a été modifiée
if ((R != R_1) || (V != V_1) || (B != B_1))
{
    sprintf(display_buffer_ligne0,"Rouge Vert Bleu");
    sprintf(display_buffer_ligne1,"%-u% %-u% %-u%",R,V,B);
    Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);
    R_1 = R; V_1 = V; B_1 = B;
}
```

(3) Partie déclarative du programme à réaliser

Le programme ci-dessus utilise deux types de variables :

- **alpha** : tableau d'octets non signés,
- **R, R_1, V, V_1, B, B_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
unsigned char R=0,R_1=0,V=0,V_1=0,B_1=0,B=0;    // intensité lumineuse
unsigned char BP;                  // Bouton-poussoir sélectionné
// Coefficient pour le réglage du rapport cyclique du signal PWM
unsigned char alpha[11] = { ???};
```



Analyse du code « Traitement » et « Partie déclarative »

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

Complétez le tableau des valeurs alpha ci-dessous en précisant l'indice i :

| i | alpha(i) |
|---|----------|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Dans l'expression `OCR1AL=alpha[R/10];` pourquoi R est-il divisé par 10 ?

Quelle est la valeur placée dans OCR1AL si le rapport cyclique affiché α est 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, PWMB et le 0V puis PB3 et le 0V.

Appel prof

Vous devez obtenir des signaux TOR dont le rapport cyclique varie par pas de 10%. Voir les exemples en annexe 1.

D2) Test avec la carte « LED haute luminosité »

Appel prof

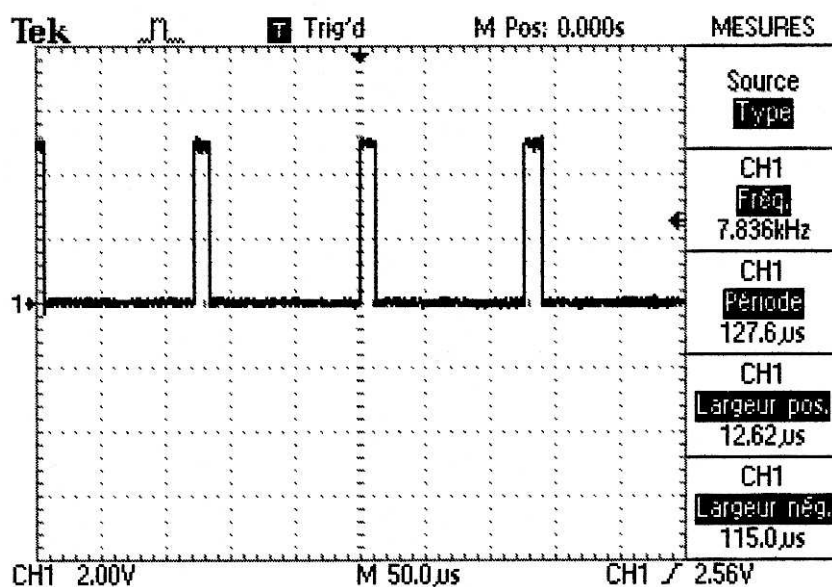
Pour connecter la carte « LED haute luminosité ».

E) Synthèse : Particularité de la commande des LED à partir du réseau DMX512

La luminosité des LED du projecteur à réaliser ne sera pas réglée par des BP mais par des valeurs numériques noté CR, CV et CB comprise dans [0,255]. Le rapport cyclique devant rester proportionnel à CR, CV et CB, modifiez le programme précédent.



Rapport cyclique = 10%



Rapport cyclique = 60%

