




<b>Fiche élève :</b>  <b>H.T.R.</b>	<b>TS SI</b>		<b>P.P.E Mini serre</b>	
	<b>Mettre en œuvre une horloge temps réel</b>			

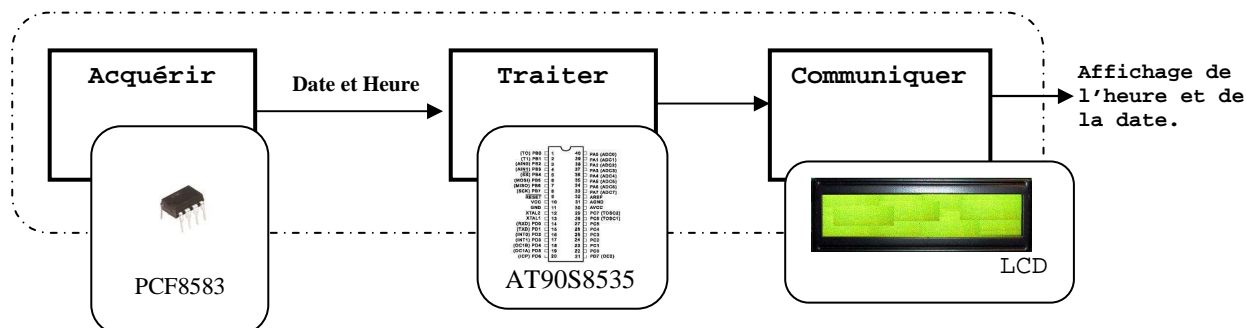
Nom :	Classe :	Groupe :
-------	----------	----------

**Objectif :** Afficher l'heure et la date sur un LCD\*.

<b>Matériels</b> Carte ATMELSSI V1 + 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 ATMELSSI. Documentation technique du circuit HTR I2C PCF8583 et de l'afficheur LCD à processeur Hitachi.

## A) Présentation

On souhaite **afficher l'heure et la date** sur un LCD. Pour cela, on se propose de mettre en œuvre une structure correspondant à la chaîne d'information donnée ci-dessous.



La fonction « Acquérir » est réalisée par une horloge temps réel PCF8583. La fonction « Traiter » est assurée par un programme implanté dans le microcontrôleur (ATMEGA8535). La fonction « Communiquer » est remplie par un afficheur LCD (processeur Hitachi).

L'ensemble des structures matérielles étant réunies sur la carte « ATMELSSI », 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 du programme avec les fonctions nécessaires à la mise en œuvre de l'horloge temps réel 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 dater les mesures faites dans la mini serre.**

\*LCD : Display Liquid Crystal

## B) Travail demandé

### Etape 1 : Création et configuration d'un projet

Lancez 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** »

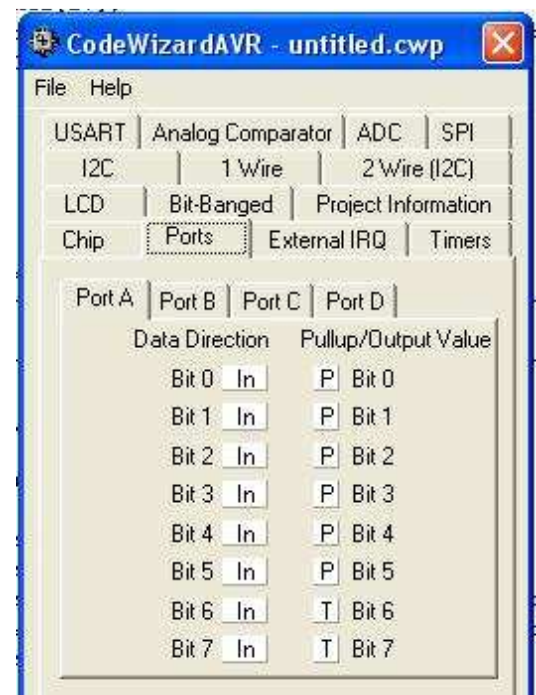
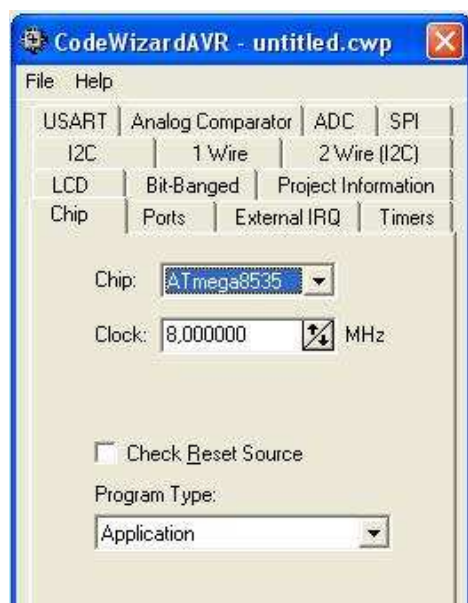


Ici « **Yes** »



#### (2) 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.



#### (3) Configuration du port A

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

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

- Configurez le Port A comme ci-contre.

#### (4) Configuration du bus I2C

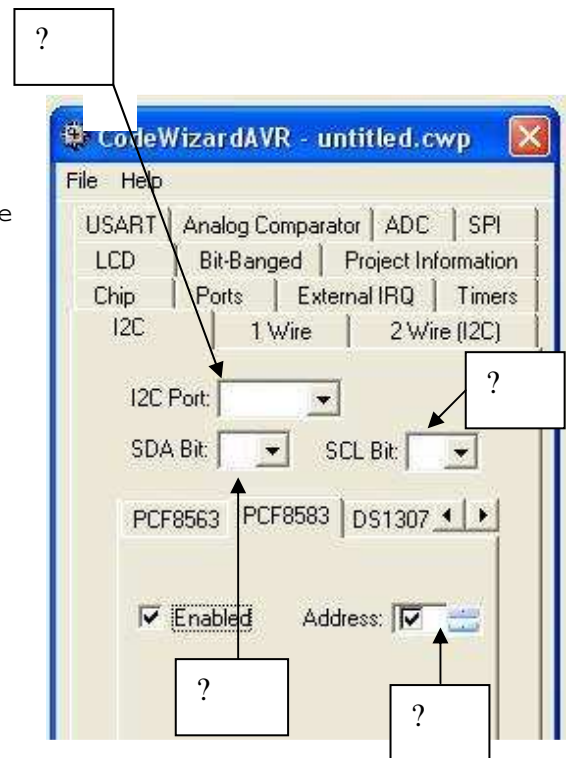
- Sélectionnez l'onglet « I2C ».

Pour que le microcontrôleur ATMEGA8535 communique avec l'horloge temps réel PCF8583, il faut configurer le bus I2C. Pour cela, il faut :

- **Identifier** la position des lignes SCL et SDA sur microcontrôleur (voir le schéma de la carte **ATMELSSI V1**) et configurer les champs « **I2C Port** », « **SDA Bit** » et « **SCL Bit** » dans la boîte de dialogue.

- Sélectionnez l'onglet PCF8583 en cochant « **Enable** ».

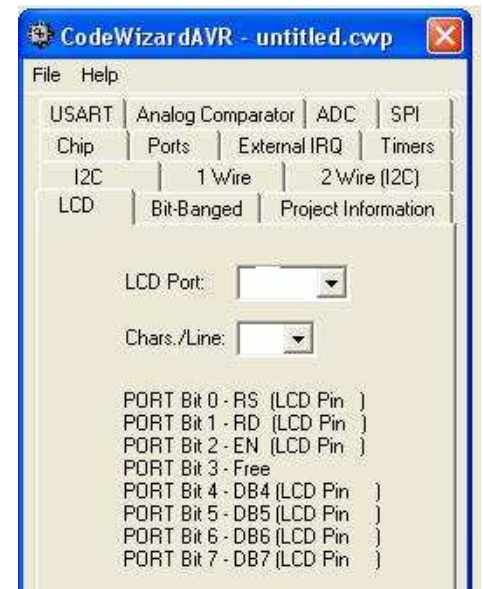
Régler le champ « **Adresse** » du PCF8583 sachant que l'adresse de ce composant est réglée avec la broche identifiée **A0**.



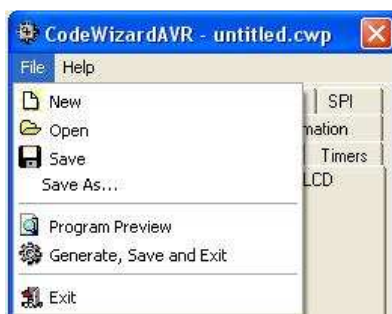
#### (5) Choix de l'affichage

- 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 ».



#### (6) Enregistrement du projet



Sélectionnez « **Program Preview** ».

Si le projet est correctement configuré, le début du fichier source du programme doit correspondre au texte ci-dessous.

```
#include <mega8535.h>

// I2C Bus functions
#asm
.equ __i2c_port=0x12
.equ __sda_bit=3
.equ __scl_bit=7
#endasm
#include <i2c.h>
```

```
// PCF8583 Real Time Clock functions
#include <pcf8583.h>

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

                                Et plus bas .....

// I2C Bus initialization
i2c_init();

// PCF8583 Real Time Clock initialization
rtc_init(0,0);

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

Fermez la fenêtre.

Sélectionnez « **Generate, save and Exit** ».

Donnez le nom **HTR** à votre projet.

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

## Etape 2 : Acquisition et affichage de l'heure et de la date

Dans cette partie, vous allez compléter la **partie déclarative**

```
// Declare your global variables here

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.

Par exemple, la lecture de l'heure fournie par l'horloge temps réel s'effectue avec la fonction :

```
rtc_get_time(adresse_HTR,&heures,&minutes,&secondes,&dizieme);
```

**adresse\_HTR** correspond à l'adresse du composant (état logique de A0). **Après la lecture, &heures,&minutes,&secondes, &dizieme** contiennent respectivement l'heure, les minutes, les secondes et les dixièmes de seconde. On accède à cette fonction par le fichier *pcf8583.h*.

La lecture de la date fournie par l'horloge temps réel s'effectue avec la fonction

```
rtc_get_date(adresse_HTR,&jour,&mois,&annee);
```

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

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

Vous devez rajouter `stdio.h` à la suite de `#include < mega8535.h >` dans le fichier source. Vous ajouterez également les librairies `ssi.h` (nécessaire pour accéder aux fonctions `Lire_BP` et `Affiche_LCD`) et `delay.h` (nécessaire pour réaliser une temporisation).

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

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

**Lire (Date et Heure)**  
**Traitement (Régler la date et l'heure)**  
**Ecrire (Afficher date et Heure)**

##### o Lecture de la date et de l'heure

La lecture de la date et de l'heure fournie par l'horloge temps réel s'effectue avec les fonctions :

```
rtc_get_time(adresse_HTR,&heures,&minutes,&secondes,&dizieme);  
rtc_get_date(adresse_HTR,&jour,&mois,&annee);
```

Complétez le fichier source C comme ci-dessous.

```
while (1)  
{  
    BP = Lire_BP();  
    // ----- Lecture et affichage de la date et de l'heure -----  
    switch (etat)  
    { // Début switch (etat)  
        case Horloge: rtc_get_time(adresse_HTR,&heures,&minutes,&secondes,&dizieme);  
                        if ((etat_1 != etat) || (secondes_1 != secondes))  
                        {  
                            rtc_get_date(adresse_HTR,&jour,&mois,&annee);  
                            sprintf(display_buffer_ligne0," %-u / %-u / %-  
i",jour,mois,annee);  
                            sprintf(display_buffer_ligne1,"%-uh / %-umn / %-us ",heures,  
minutes, secondes);  
                            Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);  
                            secondes_1 = secondes;  
                        }  
                        if (BP==ENTR) etat = Reglage_HTR;  
                        else etat = Horloge;  
                        etat_1 = Horloge;  
        break;
```

o Traitement (Réglage de la date et de l'heure)

Le réglage de la date et de l'heure s'effectue avec la fonction :

```
regle_HTR(adresse_HTR,Heure_ou_Date,H_J,mn_mois,sec_annee);
```

Complétez le fichier source C comme ci-dessous.

```
// ----- Traitement -----
case Reglage_HTR : if (etat_1 != etat)
{
    sprintf(display_buffer_ligne0,"  Reglage HTR");
    sprintf(display_buffer_ligne1,"SET>H ETR>Date ");
    Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);
}

switch(BP)
{
    // Les noms des BP sont contenus dans ssi.h
    case SET : Heure_ou_Date = 0;
               H_J = 12;
               mn_mois = 30;
               sec_annee = 0;
               regle_HTR(adresse_HTR,Heure_ou_Date,H_J,mn_mois,sec_annee);
               etat = Horloge;

               break;

    case ENTR : Heure_ou_Date = 1;
                H_J = 15;
                mn_mois = 6;
                sec_annee = 2008;
                regle_HTR(adresse_HTR,Heure_ou_Date,H_J,mn_mois,sec_annee);
                etat = Horloge;

                break;

    default:    etat = Reglage_HTR;
}
etat_1 = Reglage_HTR;
break;

} //fin du switch
} // fin du while
```

(3) Partie déclarative du programme à réaliser

Deux zones de variables doivent être complétées :

Complétez le fichier source C comme ci-dessous :

```
// Declare your global variables here
char display_buffer_ligne0[17];           // tampon ligne 0 de l'afficheur
char display_buffer_ligne1[17];           // tampon ligne 1 de l'afficheur
unsigned char heures, minutes, secondes, secondes_1, dixieme;
                                           //stockage de l'heure
unsigned char Heure_ou_Date, H_J, mn_mois;
unsigned int sec_annee;
```

```
// Declare your local variables here
unsigned char jour,mois, BP;
unsigned int annee;
unsigned char etat = Horloge, etat_1 = 2;
```

pour terminer !

```
#include <mega8535.h>
#include <ssi.h>      // contient les fonctions Lire_BP, Affiche_LCD etc..
#include <stdio.h>     // contient la fonction sprintf
#include <delay.h>     // contient la fonction delay_ms

//Etats du graphe
#define Horloge 0
#define Reglage_HTR 1

//Adresse HTR
#define adresse_HTR 0
```

## 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

*Appel prof*

Pour vérification du fonctionnement