PRESENTATION DU PROJET :

On vous demande d'étudier le concept de gestion de ventilation automatisée des boitiers de poste informatique.

On désire afficher l'heure et la date sur écran LCD à partir d'une horloge temps réel (DS1307) connectée sur bus I2C, ce qui permettra également par la suite de calculer la consommation électrique (en Wh).

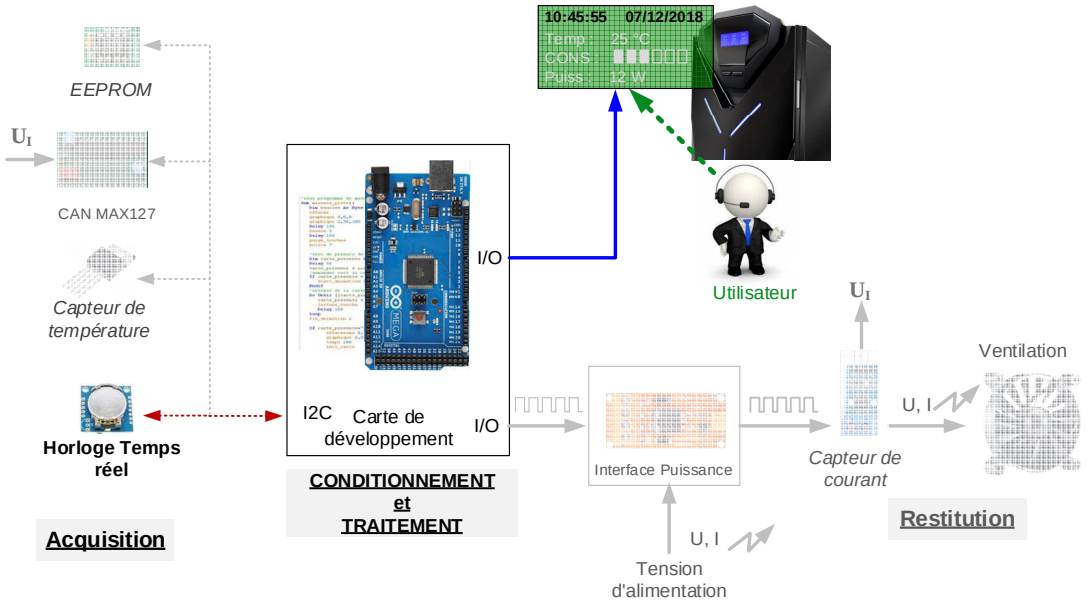
Il nous faut dans un premier temps afficher ces informations en LOCAL.

Se pose alors plusieurs problématiques auxquelles nous allons répondre à travers des différents TP mettant en œuvre le principe du bus I2C :

- *Comment mesurer la température à l'intérieur du boitier ?*
- ***Comment connaitre l'heure et la date indépendamment du PC ?***
- *Comment mémoriser les données de température ?*
- *Comment mesurer la puissance électrique consommée par le ou les ventilateurs ?*

Nous nous attacherons dans cette partie à répondre au besoin de mesure de l'heure et de la date :

- Affichage de la Date au format "JJ/MM/AAAA" et de l'heure "HH:MM"
- Modification de l'heure toute les secondes



Problématique posée : *Comment mettre en œuvre une gestion de la date et de l'heure sur un système microprogrammé ?*

L'objectif est donc d'afficher en **temps réel** les données contenues dans l'horloge électronique et de l'afficher sur écran LCD :

- Affichage sur la 1ère ligne de l'afficheur de la date et de l'heure
- Changement de l'heure toute les secondes
- Changement de la date automatiquement

Cette Activité est évaluée par revue de projet individuelle !

Étude de la programmation I²C sous Arduino avec le circuit DS1307

Q1 : Programmation Lecture de la DATE et de l'HEURE : procédure **getDateDs1307**

En fonction des instructions I²C du langage Arduino, et du chronogramme étudié ci-avant, complétez la procédure **getDateDs1307** permettant de lire les informations de l'heure à partir du circuit DS1307.

Description des fonctions I2C :

Wire.**begin**() : initialise le mode de communication I2C. A écrire une fois dans la procédure **Setup**( )

Wire.**beginTransmission**(**adress**) : débute une transmission avec un circuit I2C. Cette fonction permet de prendre la main sur le bus, et d'indiquer, par la variable **adress,** avec quel circuit l'on désire communiquer.

Wire.**requestFrom** (***adress***, ***quantity***, ***STOP***) :

Requête envoyée à l'esclave (***adress***) lui demandant de placer sur le bus I2C les données de son ou de ses registres. Le bit R/W est alors automatiquement mis à 1 (Lecture).

Quantity correspond au nombre **d'octets** à lire. ***STOP*** est un booléen (True/False) indiquant si l'on doit placer un stop après exécution de la requête pour libérer le bus.

Wire.**available**() : vérifie si les données sont disponibles sur le BUS après le lancement d'une requête. Cette fonction renvoie le nombre d'octets disponibles.

Wire.**read**() : Permet de lire un ou plusieurs octets placés par l'esclave sur le bus. Cette fonction ne peut fonctionner que si un début de transmission est effectué et si les données sont disponibles.

Wire.**endTransmission**() : Termine une transmission en cours sur le bus I2C. Cette fonction permet de libérer le bus en plaçant un STOP sur la trame.

Wire.**write**() : Permet d'écrire une donnée sur le bus. Cette fonction ne peut fonctionner que si un début de transmission est effectué. Le bit R/W est alors automatiquement mis à 0.

Remarque : on utilisera la fonction BYTE **bcd2Dec (** BYTE ***value*** **)** permettant de convertir en décimal un octet correspondant à la valeur BCD (Binaire Codé Décimal) *value* passée en paramètre. :

```
byte bcdToDec ( byte val )
{
        return ( (val/16*10) + (val%16) );
}
```

L'algorithme de la procédure **getDateDs1307** est le suivant :

Début de la transmission I2C avec adresse DS1307

Positionnement à l'adresse 0 du registre de contrôle

Requête de lecture des 7 octets du registre avec STOP final

second = conversion bcdToDec de la donnée lue en I2C
*minute = conversion bcdToDec de (donnée lue en I2C **ET** $.....)*
*hour = conversion bcdToDec de (donnée lue en I2C **ET** $.....)*
*dayOfWeek = conversion bcdToDec de (donnée lue en I2C **ET** $.....)*
*dayOfMonth = conversion bcdToDec de (donnée lue en I2C **ET** $.....)*
*month = conversion bcdToDec de (donnée lue en I2C **ET** $.....)*
year = conversion bcdToDec de la donnée lue en I2C

Temporisation de 10 ms

Fin de transmission : libération du BUS

IMPORTANT : le ET logique avec la donnée lue sur le bus I2C permet de nettoyer les bits qui ne sont pas utilisés (opérateur « & » sous Arduino).

Q2 : Programmation Écriture de la DATE et de l'HEURE : procédure **setDateDs1307**

En fonction des instructions I²C du langage Arduino et de l'étude du circuit DS1307, complétez la procédure **setDateDs1307** permettant d'écrire une date et l'heure dans le registre du circuit DS1307.

Remarque : on utilisera la fonction BYTE **decToBcd (** BYTE ***value*** **)** permettant de renvoyer un octet au format BCD (Binaire Codé Décimal) correspondant à la valeur Décimale (*value*) passée en paramètre.

```
byte decToBcd ( byte val )
{
        return ( (val/10*16) + (val%10) );
}
```

Exemple : En considérant l'heure à 12H05:56 → *heure* = 12 = %000 1100 = $0C

```
Wire.write (heure);                 //heure = %0000 1100 = $0C = "0C" ????

Wire.write( decToBcd (heure) );     // bin2bcd ( heure ) = %0001 0010 = $12 = "12"
```

L'algorithme de la procédure **setDateDs1307** est le suivant :

Début de la transmission I2C avec adresse DS1307

Positionnement à l'adresse 0 du registre de contrôle

Requête de lecture des 7 octets du registre avec STOP final

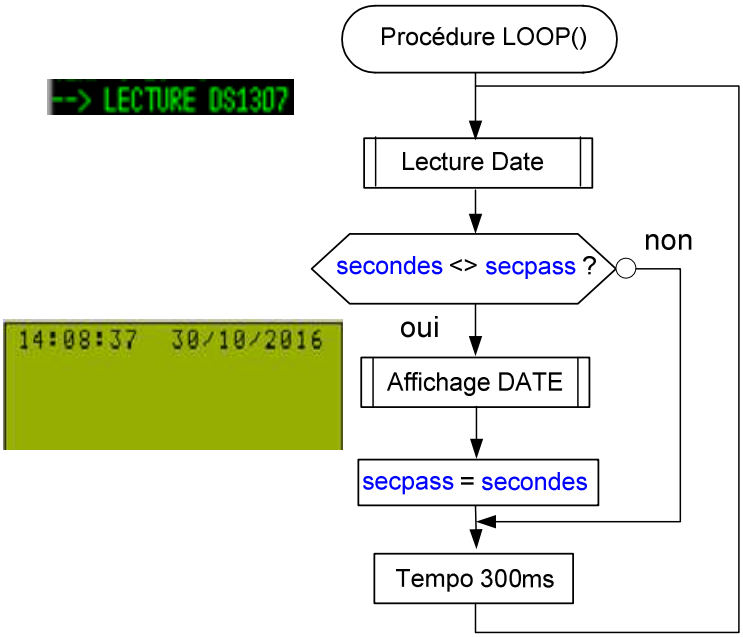
Écriture de la conversion en BCD des secondes (second)
Écriture de la conversion en BCD des minutes (minute)
Écriture de la conversion en BCD de l'heure (hour)
Écriture de la conversion en BCD du numéro du jour de la semaine (dayOfweek)
Écriture de la conversion en BCD du jour du mois (dayOfMonth)
Écriture de la conversion en BCD du mois (month)
Écriture de la conversion en BCD de l'année (year)
Écriture du registre de contrôle : Sortie OUT à l'état logique 0

Temporisation de 10 ms

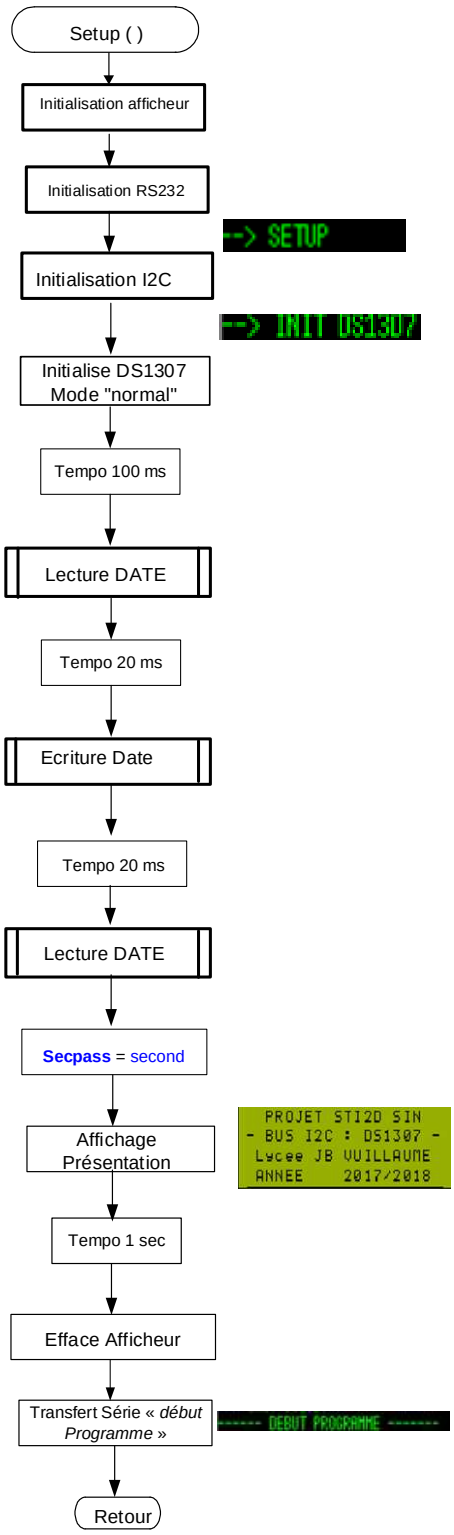
Fin de transmission : libération du BUS

♦ ORGANIGRAMMES :

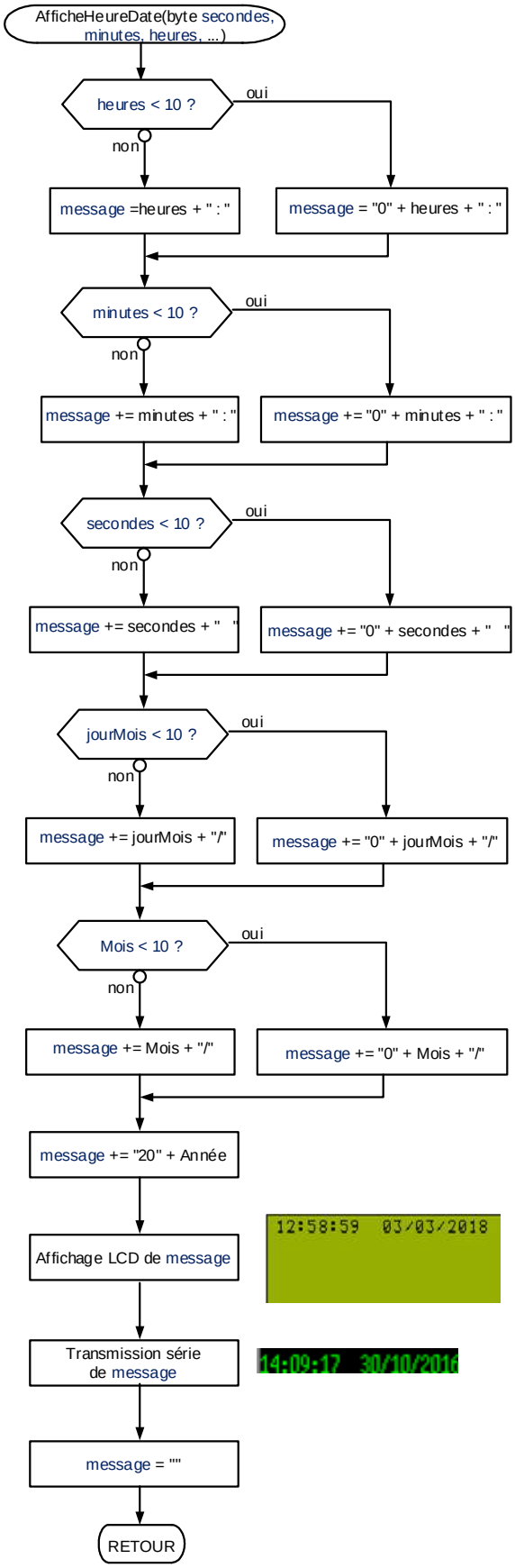
Programme PRINCIPAL (procédure LOOP) :



SP Initialisation () :



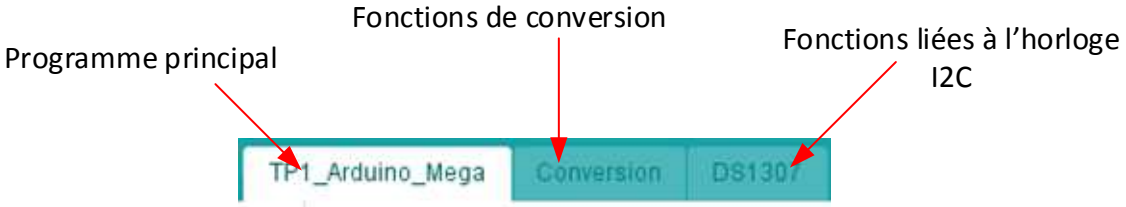
SP AfficheHeureDate (sec,min,...):



♦ REALISATION du PROGRAMME :

☞ Ouvrez le programme « *TP1_Arduino_Mega.ino* » contenu dans le dossier de ressources « *\programme* »

Celui-ci est organisé en Onglets :

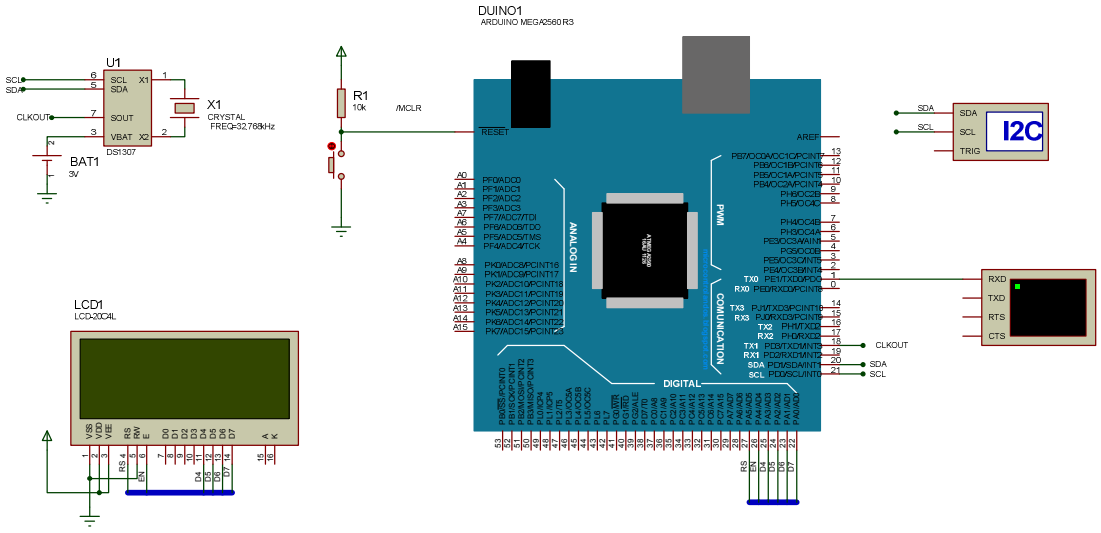


☞ Complétez les différentes parties du programme en vous aidant du cahier des charges, à savoir :

- Procédure d'initialisation **setup**()
- Programme principal **loop**()
- Fonction **AffichageHeureDate** (...)
- Fonction **getDateDs1307**(...)
- Fonction **setDateDs1307**(...)

♦ SIMULATION et MISE AU POINT du PROGRAMME :

☞ Ouvrez le fichier de simulation **Proteus ISIS** « *TP1-HORLOGE-I2C.DSN* » contenu dans le dossier de ressources « \simulation »



☞ Placez le programme ***TP1_Arduino_Mega.ino.HEX*** dans le modèle de simulation Arduino, et simulez le montage. Mettre au point le programme afin de répondre au cahier des charges.

TRAVAILLEZ PAR ÉTAPE !!! Utilisez les opérateurs /* et */ pour mettre en commentaire les fonctions ou parties de programme en attentes d'écriture.

Utilisez le débogueur I2C pour analyser le contenu des trames !