http://www.didel.com/ **info@didel.com**

www.didel.com/OledDS1307.pdf

Expériences avec le circuit horloge DS1307

Programmes sous www.didel.com/OledDS1307.zip

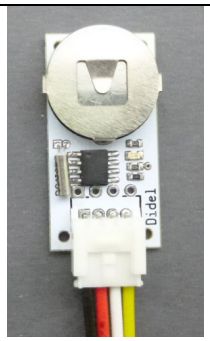
Remarque préliminaire

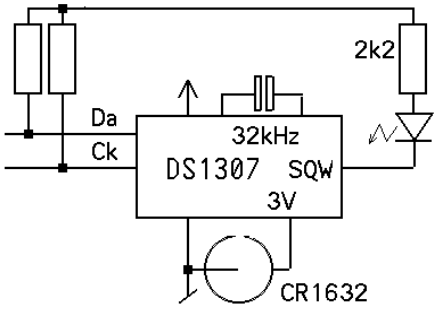
Ce document est un excellent complément de formation à la programmation en C sous Arduino. Un débutant verra une application intéressante et apprendra à mieux programmer. Un connaisseur verra comment insérer un circuit horloge dans sa prochaine application. Pour l'enseignant. ce document est une base pour préparer un module d'initiation à la programmation.
Avoir suivi le document " Expériences avec le Oled SSD1306" n'est pas nécessaire si le but est seulement de comprendre le DS1307 et d'afficher l'heure de la façon la plus simple. Afficher l'heure de façon originale sur un écran de 126x64 suppose évidemment une bonne connaissance des primitivres graphiques.

1 Module Ds1307

Le circuit DS1307 a une interface de commande I2C. Une pile maintient l'heure et le quartz 32 kHz donne une précision de quelques secondes par jour.
La documentation du DS1307 se trouve sous http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

Le module utilisé a un connecteur Grove et 4 pins au pas de 2.54mm. On reconnait sur le schéma les résistances demandées par I2C, 4k7 préférable si on utilise plusieurs unités I2C. La pile continue à alimenter les compteurs de temps si le circuit n'est pas sous tension (3 à 5V).
Ce circuit consomme moins d'un mA, il peut donc être aussi alimenté par des pins Arduino libres.





Des modules équivalents existent sur le marché et on trouve sous Arduino plusieurs exemples d'utilisation avec la librairie Wire (2,5k), l'adresse 7 bits du DS1307 est 104 en décimal. L'affichage se fait avec le terminal série d'Arduino.

On va tester ici le DS1307 en utilisant une librairie I2C minimale (0.4k) avec des fonctions simples à comprendre, et profiter d'un affichage Oled pour afficher les résultats.

2 Circuit horloge DS1307

Dans le circuit, l'heure est mémorisée dans des registres. L'adresse I2C est suivie par une adresse locale qui sélectionne l'un de 8 registres (Table 2) dans lequel on va écrire.

Table 2. Timekeeper Registers

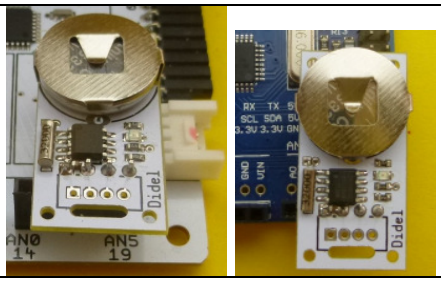
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM
		24	PM/ AM							00–23
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

Humm, Cela semble compliqué! Allons par étape.

3 Connexion du module

Le module DgDs1307 a deux connecteurs. Le connecteur Grove permet de se brancher directement sur la carte Arduino, ou sur un shield bourré de connecteurs. Avec les 4 pins au pas de 2.54mm, on peut se connecter sur un breadboard. Notre solution est la plus simple: le module est inséré directement sur les pins Arduino 16 (Gnd) 17(5V) 18(Sda) 19(Scl).



Sur la carte Arduino, il y a trop peu de pins d'alimentation. Utiliser des signaux comme alimentation n'est pas connu. Il faut naturellement respecter des contraintes: courant maximum de quelques mA, condensateur de découplage s'il y a des pointes de courant. Une sortie du processeur AVR328 a une résistance interne d'une dizaine d'Ohm (c'est connu qu'il ne faut pas demander plus de 30mA pour allumer une LED).
Pour déclarer dans le setup que des pins sont au Gnd ou +3V, il y a la façon Arduino et la façon C que nous utiliserons par la suite.

Gnd et +5V sur les pins 16 et 17

```
int Gnd 16
int Vcc 17
pinMode (Gnd,OUTPUT);
pinMode (Vcc,OUTPUT);
digitalWrite (Gnd,LOW);
digitalWrite (Vcc,HIGH);
```

Gnd et +5V sur le PORTC bits 2 et 3 (pins 16,17)

```
#define bGnd  2
#define bVcc  3
DDRC   |= (1<<bGnd)+(1<<bVcc) ;
PORTC  |= (1<<bVcc) ;
PORTC &= ~(1<<bGnd) ;
```

4. Test de la sortie SQW

Comme on le voit sur le schéma, la sortie SQW pilote une Led.
Le comportement de cette LED dépend du registre "control" (adresse 7) de la Table2. On peut allumer, éteindre, et clignoter cette LED avec une vitesse qui dépend des bits RS1 RS0. Si on active SQWE avec les autres bits à zero, on lance un clignotement à 1 Hz, c'est ce que l'on va faire. Le mot binaire qui correspond est 0b00010000 = 0x10 que l'on prèfère écrire en C 1<<4, évidemment en utilisant la déclaration cachée SQWE=1, donc 1<<SQWE qui est le plus clair.

Le programme, une fois déclaré le câblage et l'adresse du 1307, transfére l'information par I2C. Le détail des instructions est caché en appelant une librairie universelle.

Arduino	Action	Librairie DS1307.h
`Wire.beginTransmission(Adr1307);`	commencer le transfert	`Start1307();`
	sélectionner le 1307 en écriture	`Write1307(Adr1307);`
`Wire.write(byte(Control));`	choisir le registre "control"	`Write1307(Control);`
`Wire.write(byte(0x10));`	l'initialiser	`Write1307(1<<SQWE);`
`Wire.endTransmission();`	fin du transfert	`Stop1307();`

Cette séquence peut être remplacée par une fonction. Ce que l'on veut, c'est initialiser le registre control du DS1307. La librairie est alors spécialisée, plus facile à utiliser.
On écrira simplement `WriteReg1307 (Control, 0x10) ;`

Le fichier DS1307.h contient les fonctions de communication entre les DS1307 et le microcontrôleur. Ce fichier définit aussi la valeur numérique de quelques mots clé, comme Control qui vaut 7 et SQWE qui vaut 4.
Donc, le programme à tester est

```
//Test1307Cli
#include "DS1307.h"
void setup() {
  SetupSD1307();
}
void loop () {
  WriteReg1307 (Control, 1<<SQWE) ;
  while (1) {}  // la led doit maintenant clignoter
}
```

Petit exercice pour plus tard: Si, dans les programmes suivants, vous ne modifiez pas le registre de commande 7, la led va-t-elle continuer à clignoter? Et si vous déconnectez et reconnectez? L'information a été mémorisée grâce à la pile, ou bien il y a une EEPROM interne. Comment expérimenter pour savoir?

5. Principe d'accès à l'information

Il faut voir le circuit 1307 comme une commode que l'on accède avec son adresse I2C. Les registres sont des tiroirs dans lesquels on trouve la valeur de l'heure (secondes minutes, etc.). L'adresse locale est le numéro du tiroir. Pour écrire les minutes par exemples, il faut "ouvrir" son tiroir en envoyant son numéro, valeur 1, puis envoyer la valeur codée des minutes.
De même pour lire un registre, on ouvre le tiroir et on lit.

Le tableau 2 du fabricant, pour le registre 2, est difficile à décoder sans aller lire les spécifications détaillées:

> Bit 6 of the hours register is defined as the 12-hour or 24-hour mode-select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 to 23 hours).

Donc pour afficher sur 24 heures, le registre 2 contient 0 0 d d u u u u, avec dd uuuu en décimal codé binaire. par exemple pour 23 heures 0 0 1 0 0 0 1 1 en hexa 0x23.
Ecrire dans un registre se fait avec la fonction WriteReg (r,val); qui a 2 paramètres: le no du registre (mais on va donner un nom pour être plus clair) et la valeur, exprimée en BCD, c'est-à-dire en hexadécimal. comme pour l'heure.
A la mise en route, il faut initialiser les 7 registres. Pour ajuster l'heure après quelques mois, il suffit d'intervenir sur les registre 1 et 2 (si ce n'est pas vers minuit).
Les instructions correspondantes s'écrivent

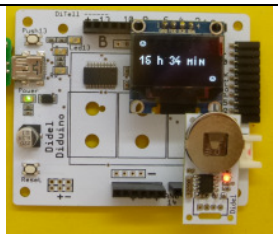
```
WriteReg (Seconds,0);
WriteReg (Minutes,0x12);
WriteReg (Hours,0x16);
WriteReg (Day,0x2);     // 1 lundi 2 mardi
WriteReg (Date,0x27);
WriteReg (Month,0x05);
WriteReg (Year,0x17);
```

Ces instructions sont dans le programme TestOled1307IniTime.ino

On peut écrire un programme qui demande ces valeurs par le terminal série, ou un clavier. On peut connecter un module DCF77 avec son programme de décodage et recevoir l'heure exacte par radio. Un module GPS donne aussi l'heure. Mais alors, à quoi sert le 1307?
Pour lire et afficher l'heure, on peut procéder de façon similaire avec la fonction
val= ReadReg(r);
Comme on l'a vu pour l'heure, les registres contiennent du décimal codé binaire, qui est de l'hexadécimal dans lequel les "chiffres" A B C D E F ne sont pas utilisés.
Il ne faut pas convertir ces nombres en décimal ! Avec le terminal Arduino, si l'heure en exemple plus haut, 23 heures, est dans la variable hh, Serial.print (hh); va afficher 35. Il faut écrire Serial.print (hh,HEX); pour voir 23. Pour la librairie Adafruit sur Oled, c'est le même principe.
Donner des noms explicites nous semble plus clair Dec8(hh); convertit la valeur en mémoire en décimal, Hex8(hh); affiche le binaire en mémoire en hexadécimal.

6 Affichage sur le Oled

Il faut maintenant pouvoir afficher des données.
Le Oled se branche sur les pins 4,5,6,7 (PortD) et sa librairie a été expliquée sous www.didel.com/Oled1306.pdf
Comme premier exercice, affichons l'heure, c'est-à-dire heure et minutes.
Les textes sont écrits caractère par caractère, cela ne faut pas la peine de créer une table.



Le programme est alors

```
//Test1307LitSecondes.ino On lit et affiche les secondes et minutes
#include "OledPix.h"
#include "DS1307.h"
 void setup(){
  SetupOledPix();
   SetupDS1307();
}
```

```
byte ss,mm;
void loop(){
  ss = ReadReg1307(Seconds); // on lit le DS1307
  mm = ReadReg1307(Minutes);
  LiCol(0,0); Sprite(smile); // on positionne le curseur
  LiCol(2,0); Hex8(mm); Car('h'); Car(' ');
  LiCol(4,0); Hex8(ss); Car('m'); Car(' ');
}
```

On peut mettre des gros caractères, mais il faut pour les positionner se souvenir qu'ils occupent aussi la ligne en dessus. Les instructions qui changent deviennent:

```
LiCol(0,0); Sprite(smile);
LiCol(3,0); BigHex8(mm); Big ('h'); Big (' ');
LiCol(2,0); BigHex8(ss); Big ('m'); Big (' ');
```

On peut aussi tester la fonction `DoubleH();` avant de décider comment faire une jolie mise en page. Avec si peu de pixels, ce n'est pas si facile. Comme on le voit, chaque élément doit être positionné.

Un petit effort de disposition a été fait avec le programme TestOled1307ReadTimeSlash.ino On voit que chaque élément doit être placé selon ses coordonnées.	

```
 byte ss,mm,hh,dd,mo,yy;
void loop(){
  LiCol(0,0); Sprite(smile);
  LiCol(7,118); Sprite(smile);

  while(1){
   ss=ReadReg(Seconds);
   mm=ReadReg(Minutes);
   hh=ReadReg(Hours);
   dd=ReadReg(Date);
   mo=ReadReg(Month);
   yy=ReadReg(Year);
 //   LiCol(2,40); BigHex8(ss); Big('s');Big(' ');
    LiCol(2,20); BigHex8(hh);
    LiCol(2,40); Big(':');
    LiCol(2,50); BigHex8(mm);
    LiCol(5,10); BigHex8(dd);
    LiCol(5,35); Big('/');
    LiCol(5,50); BigHex8(mo);
    LiCol(5,75); Big('/');
    LiCol(5,90); BigHex8(yy);
     Hline(25);
    delay(200);
  }
}
```

7 Compatibilité AdaFruit/GFX

Pour des effets graphiques intéressants, il faut utiliser la librairie OledMap.h ou la librairie Adafruit, qui offre plus d'éléments géométriques et des caractères de taille 1,2,3,4.

Cette librairie oblige de permuter le DS1307 et l'affichage. Il faut de plus câbler Sda sur le pin 18/A4 et Scl sur la pin 19/A5 (l'insertion directe sur le PORTC n'est pas possible).
Voir AdafOledDs1307.ino Cette partie sera développée avec la doc OledMap.

Exemples trouvés sur le web.