TP 3 : Mise en œuvre d'un capteur de température I2C DS1621.

Encadrants : Hélène LEYMARIE // Thierry PERISSE

Objectifs :

Afficher sur un LCD la valeur de la température donnée par un capteur I2C (DS1631)

Savoir faire : (pré-requis)

Savoir utiliser le logiciel "mikroC PRO for PIC" permettant de programmer un microcontrôleur PIC16F877 avec un langage de haut niveau (langage C).

Savoir utiliser le logiciel de simulation de schéma électrique "PROTEUS"

Savoir comprendre une communication I2C

Outils logiciels :

PROTEUS LIEN LABCENTER

mikroC PRO for PIC LIEN MIKROE

Procédure Téléchargement PROTEUS demo

Procédure Téléchargement mikroC PRO for PIC

Matériels hardwares utilisés

Aa Composants	Références
Capteur de température I2C	DS1631
Microcontroleur	PIC16F877
Afficheur LCD	LCD 2×16
LED	LED
Bouton poussoir	BP

TP3 : Mise en œuvre d'un capteur de température I2C.(4h)

Projet n°1 : LED

Projet n°2 : LED_BP

Projet n°3 : LED_BP_LCD

Projet n°4 : LED_BP_LCD_DS1631

Projet n°5 : LED_BP_LCD_DS1631_Thermostat

Projet n°6 : LED_BP_LCD_DS1631_Thermostat_bis

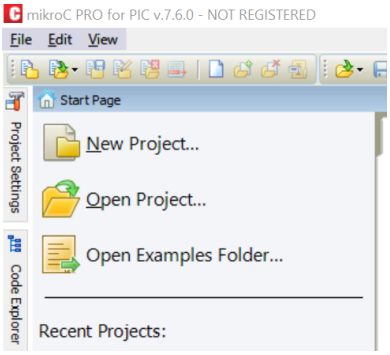
Projet n°1 : LED

Prise en main des logiciels mikroC PRO for PIC (et PROTEUS)

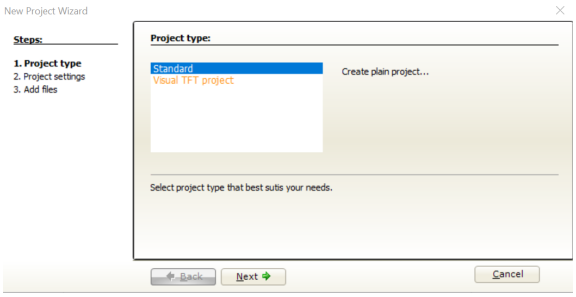
Cahier des charges :

Faire clignoter une LED (ON 1s OFF 1s)(sortie 7 du PORTD)

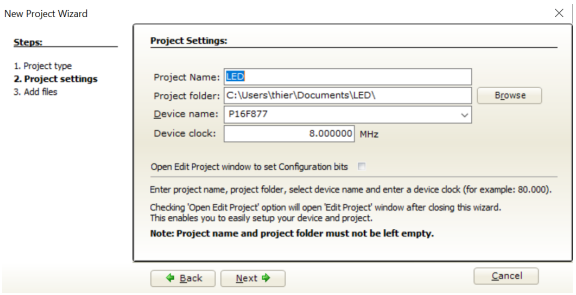
Prise en main de mikroC PRO for PIC



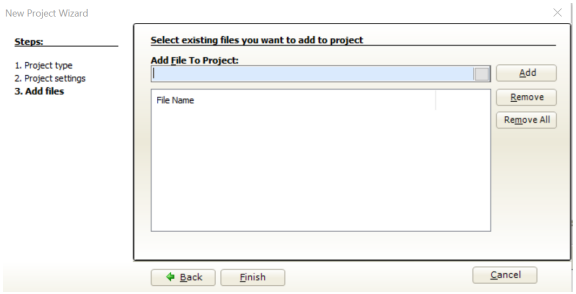
Créer un nouveau projet



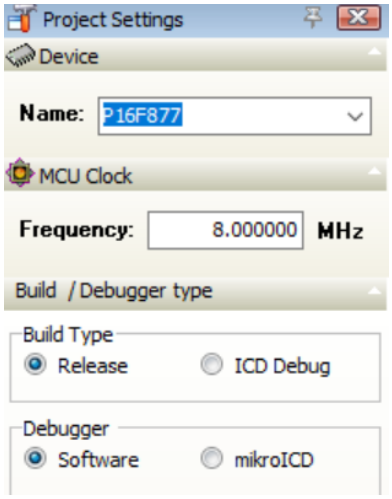
Choisir Projet standard



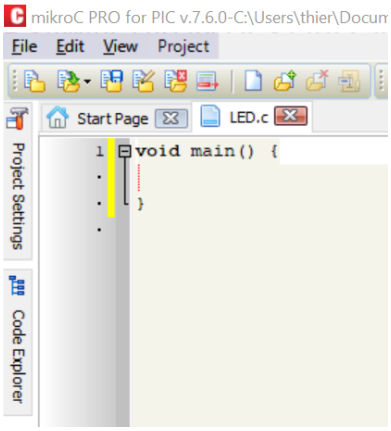
Choisir le nom du projet LED, le répertoire de sauvegarde .../LED/, le microcontrôleur 16F877, l'horloge 8Mhz



Pas de fichier à ajouter au projet



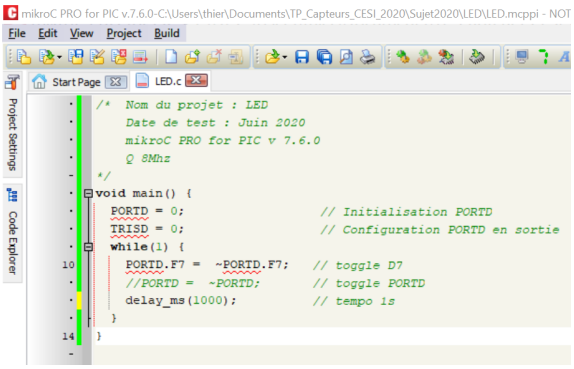
Choisir les paramètres de Project Settings



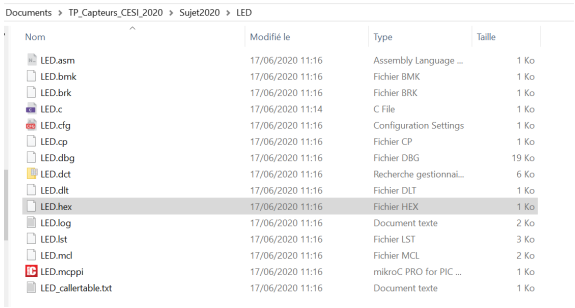
Projet vide

```
/*  Nom du projet : LED
    Date de test : Juin 2020
    mikroC PRO for PIC v 7.6.0
    Q 8Mhz
```

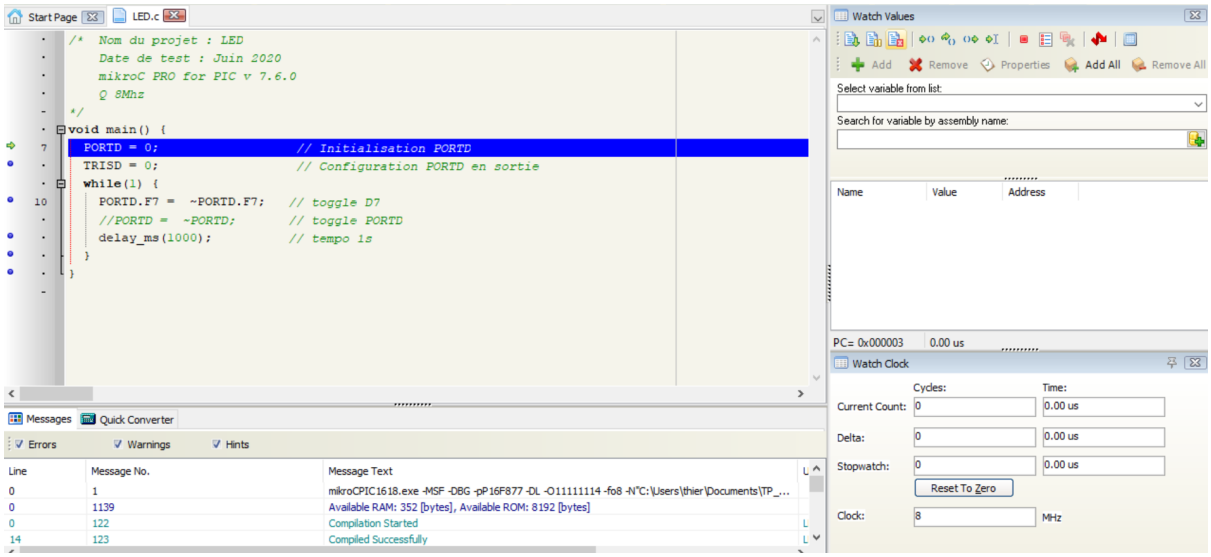
```
*/
void main() {
  PORTD = 0;                  // Initialisation PORTD
  TRISD = 0;                  // Configuration PORTD en sortie
  while(1) {
    PORTD.F7 =  ~PORTD.F7;   // toggle D7
    //PORTD =  ~PORTD;        // toggle PORTD
    delay_ms(1000);           // tempo 1s
  }
}
```



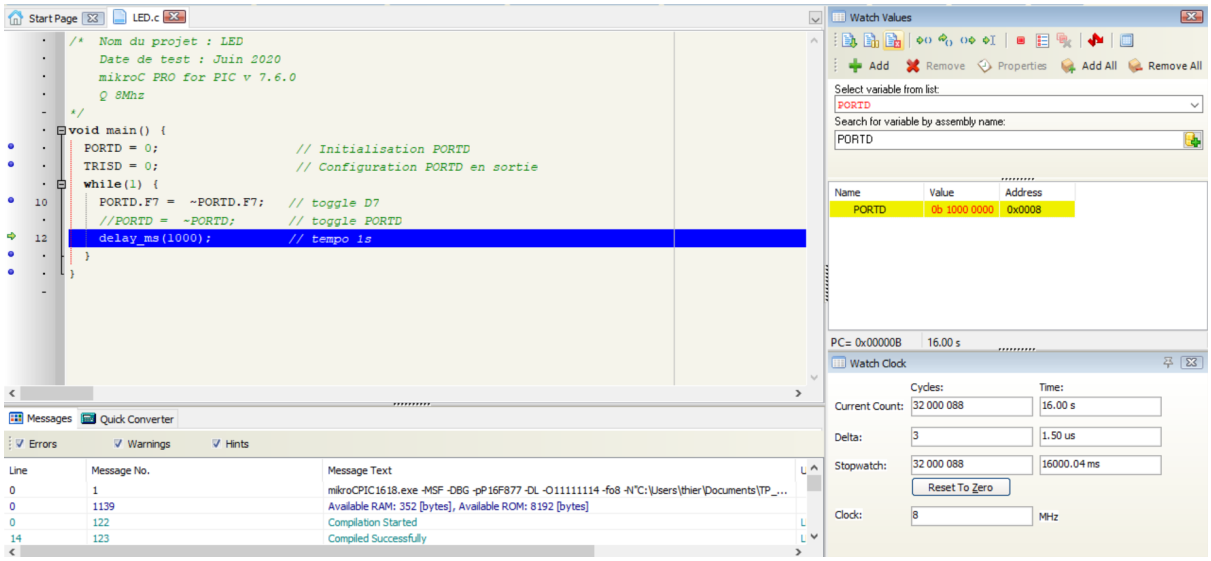
Copier / Coller du programme ci-dessus LED Compiler le projet "Build", vérifier dans la fenêtre "Messages" il y a des erreurs.



Regarder maintenant dans le répertoire LED les fichiers qui ont étés crées (en particulier "LED.hex" qui va nous servir pour programmer le microcontrôleur dans PROTEUS)



Faire avancer le programme pas à pas et vérifier le bon fonctionnement en regardant le contenu du PORTD



Prise en main de PROTEUS

Lancer PROTEUS Professionnel

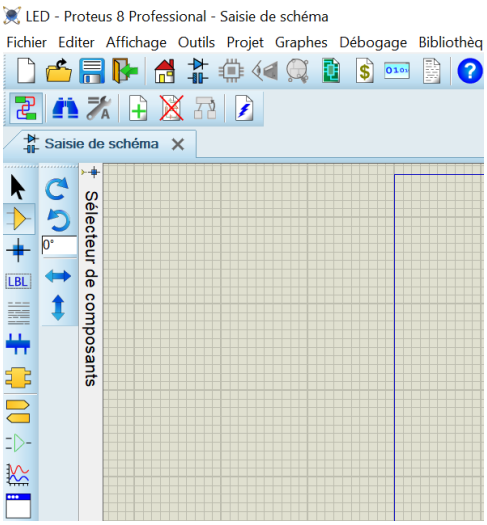
Dans "Fichier" sélectionner "Nouveau projet"

Choisir Nom : "LED.pdsprj"

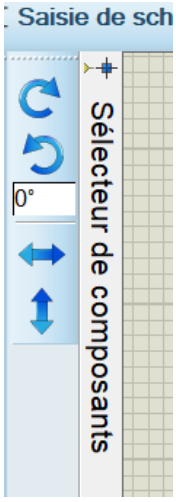
Choisir Chemin : " ...\PROTEUS\LED" il faut créer un répertoire "PROTEUS" et un sous répertoire "LED"

"suivant" Choisir Gabarit "DEFAULT"

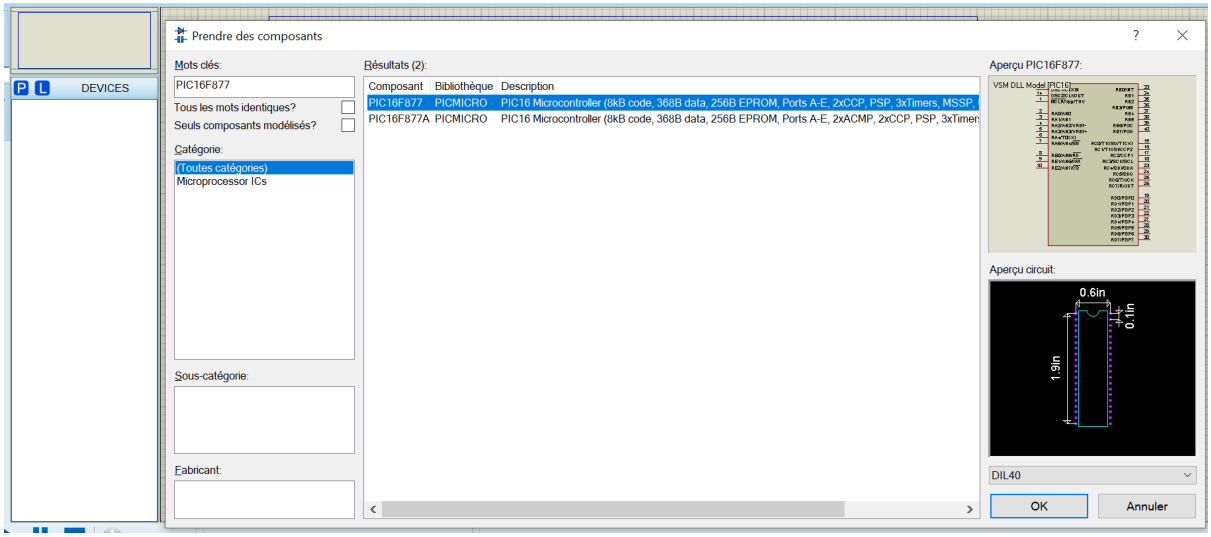
"Pas de projet firmware" puis "suivant" et "Terminer"



Projet "LED" vierge

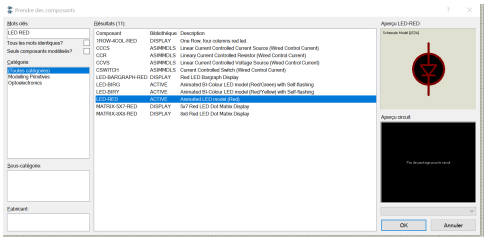


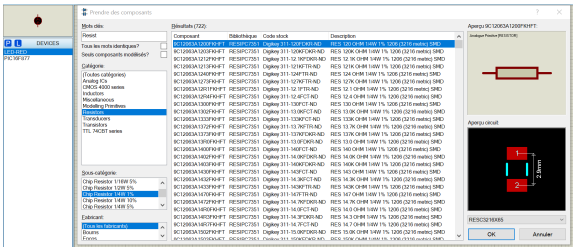
Recherche composants



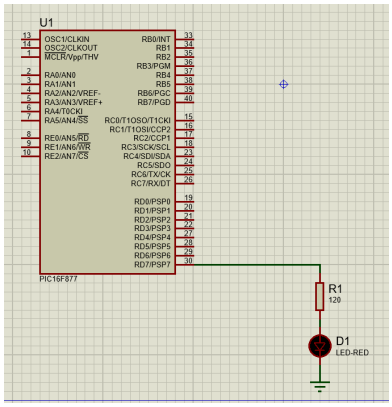
Recherche dans mots clés "PIC16F877" choisir le microcontrôleur désiré dans résultats et "ok"

Faire la même chose pour rapatrié une LED "LED RED" et une résistance "résistors"





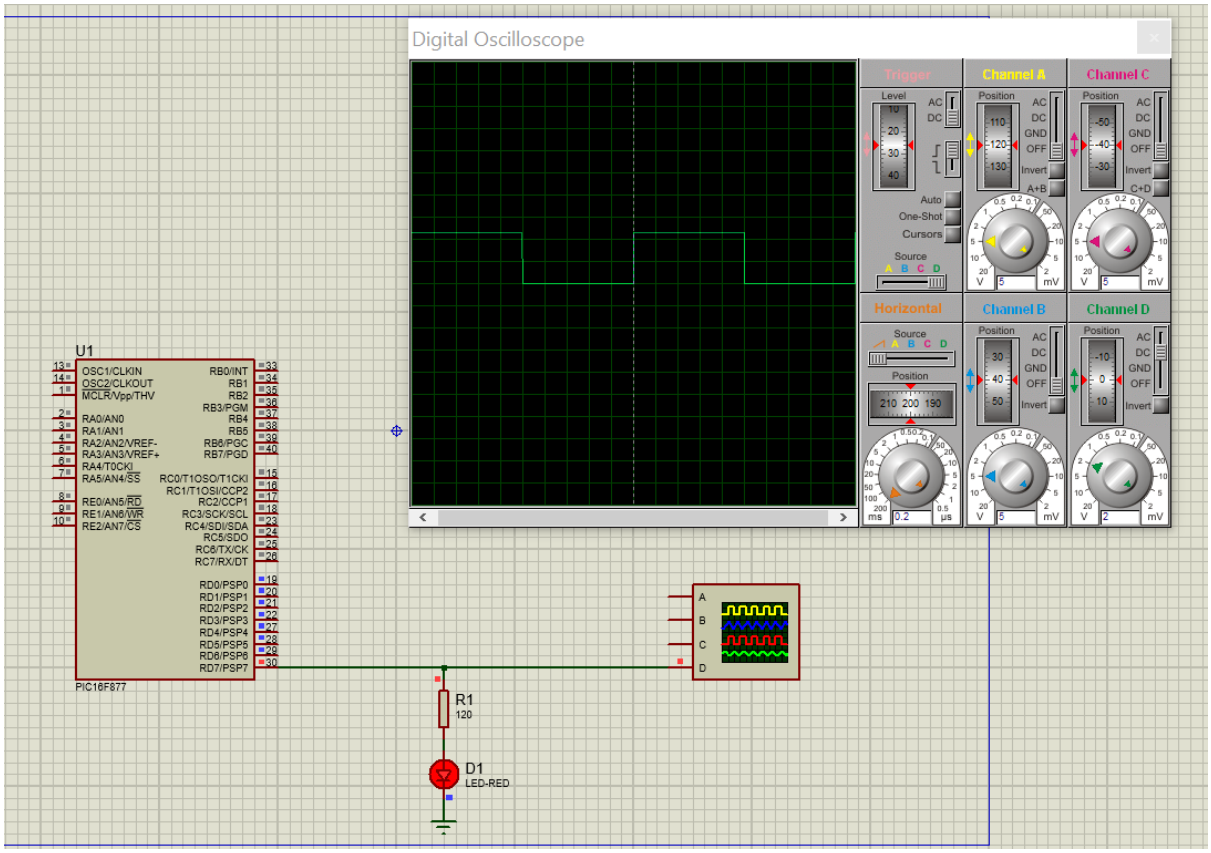
Rapatrier les 3 composants (PIC16F877, LED RED, R) (+ la masse) sur la feuille vierge et réaliser le câblage.



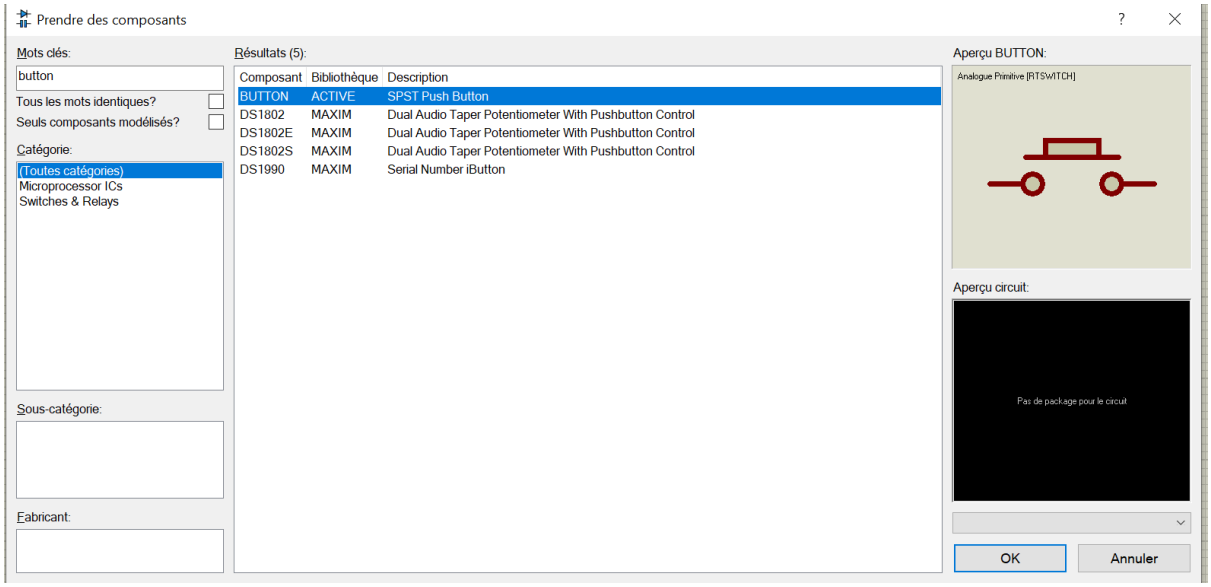
Sélectionner le microcontrôleur 16F877 (en cliquant dessus), puis click droit et choisir "Edit Properties", vérifier l'horloge (8MHz), et choisir dans Program File le fichier à charger dans le micro ici "LED.hex".

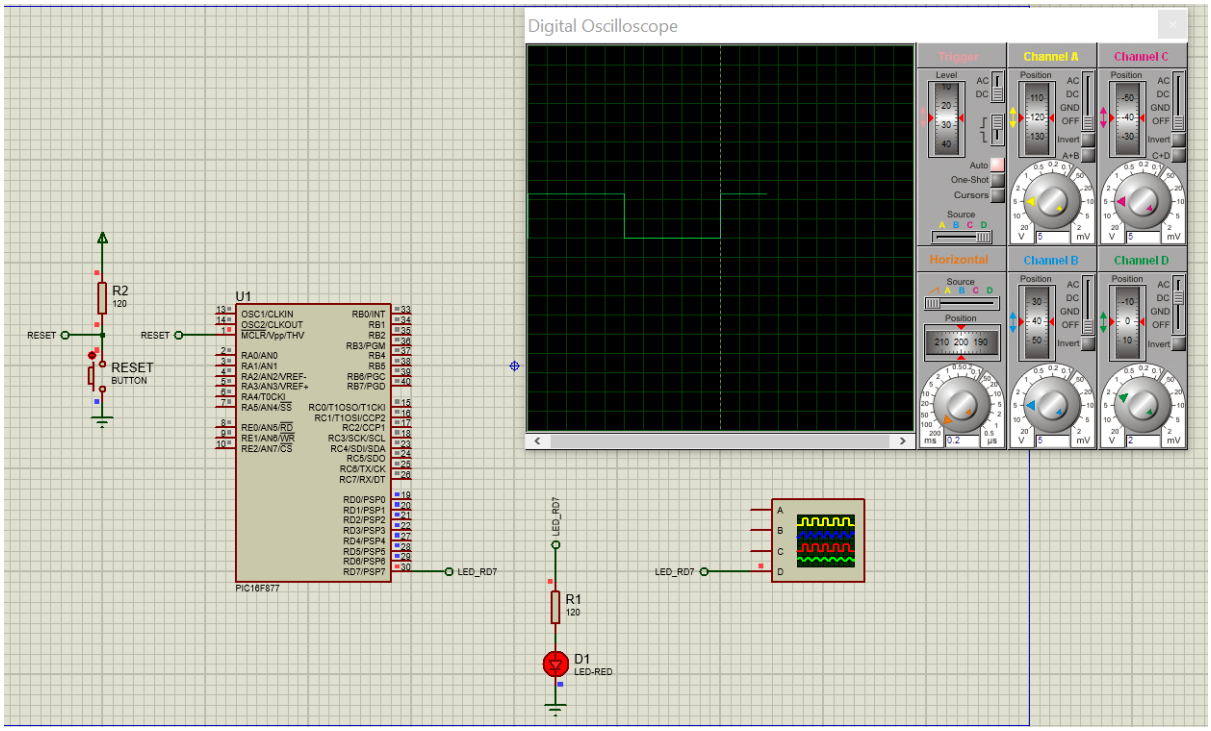
Sélectionner le microcontrôleur 16F877 (en cliquant dessus), puis click droit et choisir "Edit Properties", vérifier l'horloge, et choisir dans Program File le fichier à charger dans le micro ici "LED.hex".

Lancer la simulation et vérifier le bon fonctionnement du montage (clignotement de la LED), vérifier le bon timing sur l'oscilloscope en utilisant les curseurs.



Avant de passer au projet suivant rajouter un bouton poussoir de reset "RESET" et arranger le câblage en donnant des noms sur les potentiels utilisés (LED_RD7 et RESET).





Tester le RESET en appuyant sur le bouton poussoir "RESET"

Pour éviter d'avoir tout le projet suivant à recâbler Enregistrer le projet LED sous un répertoire ...\PROTEUS\LED_BP avec le nom "LED_BP.pdsprj"

Projet n°2 : LED_BP

Cahier des charges :

Si BP OFF Faire clignoter une LED (ON 1s OFF 1s), si BP ON LED OFF. LED sortie 7 du PORTD et BP entrée 6 du PORTD.

LED_BP sur mikroC PRO for PIC

Créer un nouveau projet LED_BP dans un nouveau répertoire ../LED_BP/

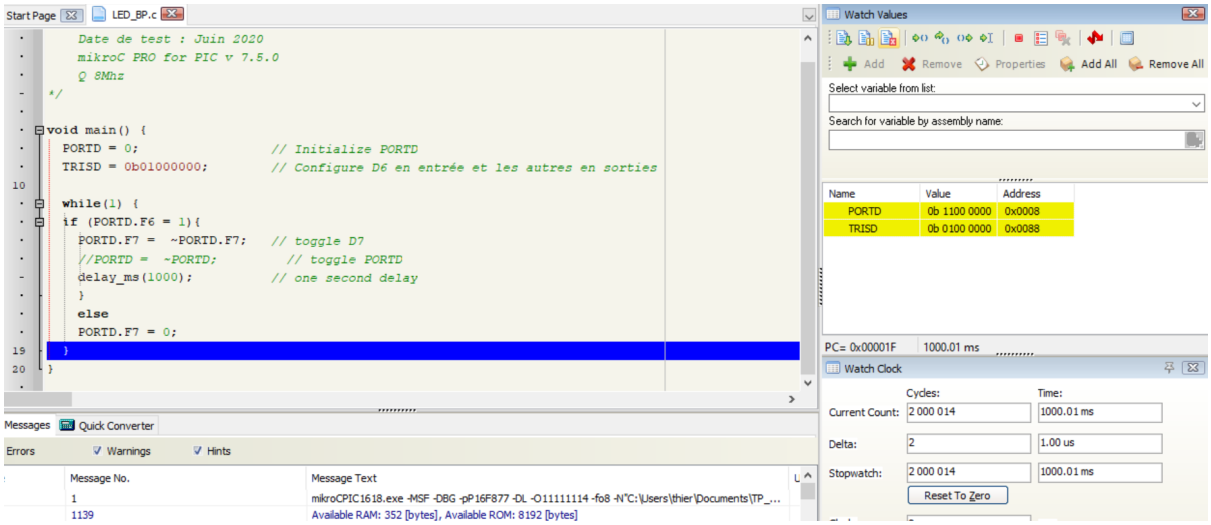
Récupérer le fichier LED_BP.c

Compiler le projet LED_BP

Run et Start Debuger

Dans la fenêtre "watch Values" vérifier le bon fonctionnement du programme en regardant le contenu du PORTD et de TRISD.

Dans la fenêtre "watch Clock" Sur 1 passage dans la boucle vérifier la tempo.



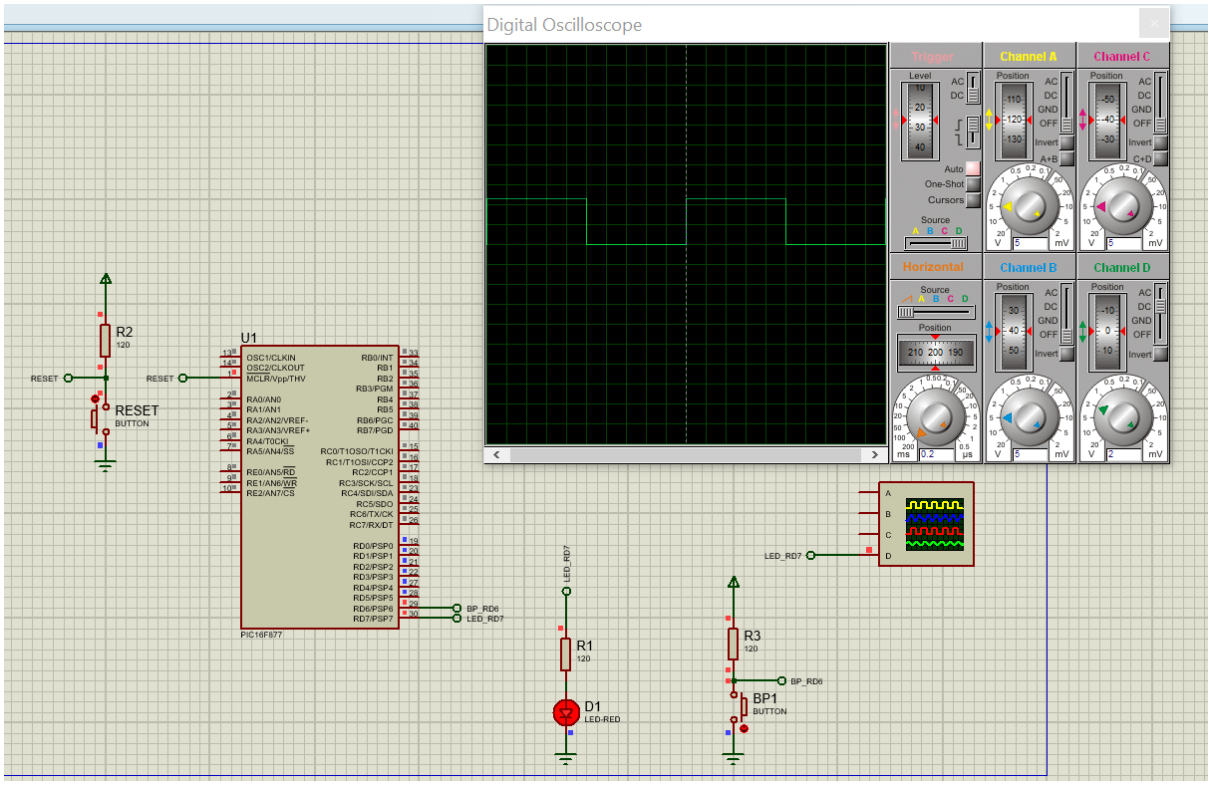
LED_BP sur Proteus

Compléter le câblage avec BP1 relié à la broche 6 du PORTD

Charger le microcontrôleur PIC16F877 avec LED_BP.hex

Lancer la simulation et vérifier le bon fonctionnement du montage :

- que se passe t'il lorsque BP1 est relâché,
- que se passe t'il lorsque BP1 est appuyé,
- que se passe t'il lorsque RESET est appuyé.



Projet n°3 : LED_BP_LCD

Cahier des charges :

Ecrire "votre prénom" sur la première ligne du LCD, écrire "votre nom" sur la deuxième ligne du LCD, si BP OFF Faire clignoter une LED (ON 1s OFF 1s), si BP ON LED OFF. LED sortie 7 du PORTD et BP entrée 6 du PORTD.

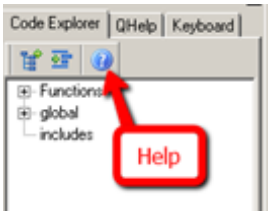
LED_BP_LCD sur mikroC PRO for PIC

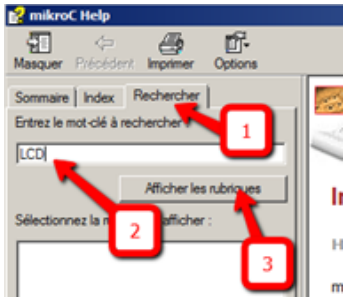
Créer un nouveau projet LED_BP_LCD dans un répertoire ../LED_BP_LCD/ et récupérer le fichier "LED_BP_LCD.c"

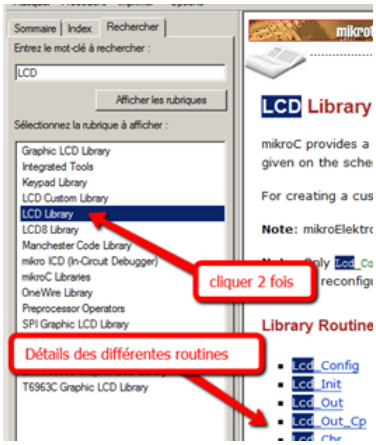
Regarder les routines pilotant le LCD (cliquer sur "Help" ou "?" (bleu)

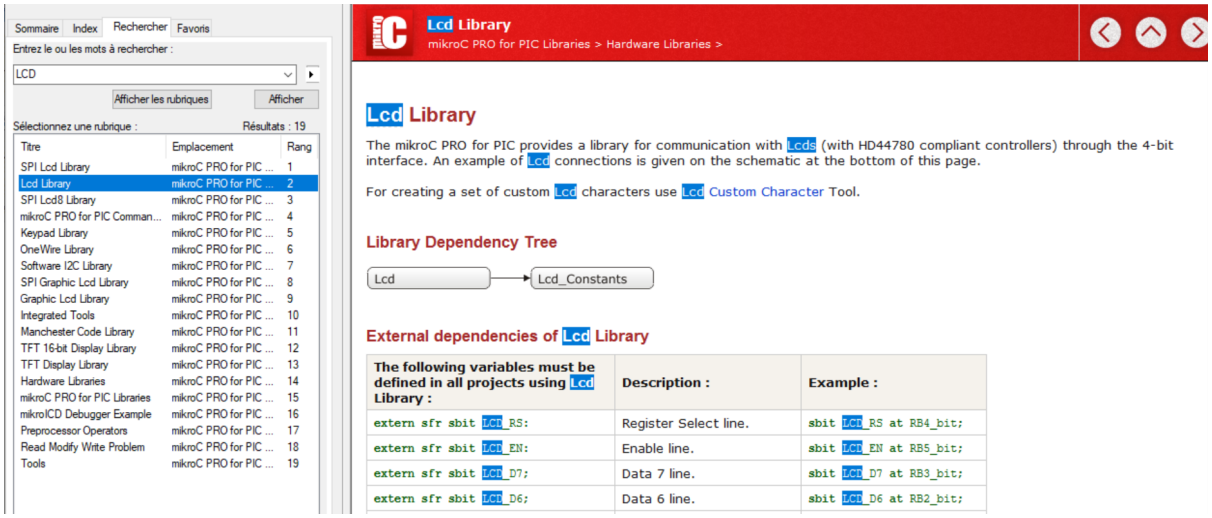
Rechercher, taper "LCD" comme mot à rechercher et valider

Sélectionner la rubrique "Lcd Library"

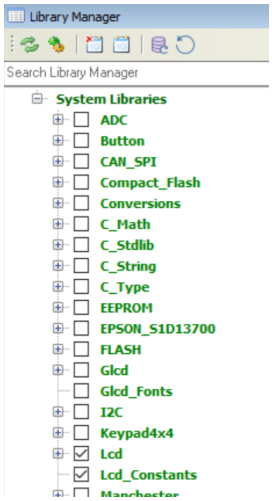








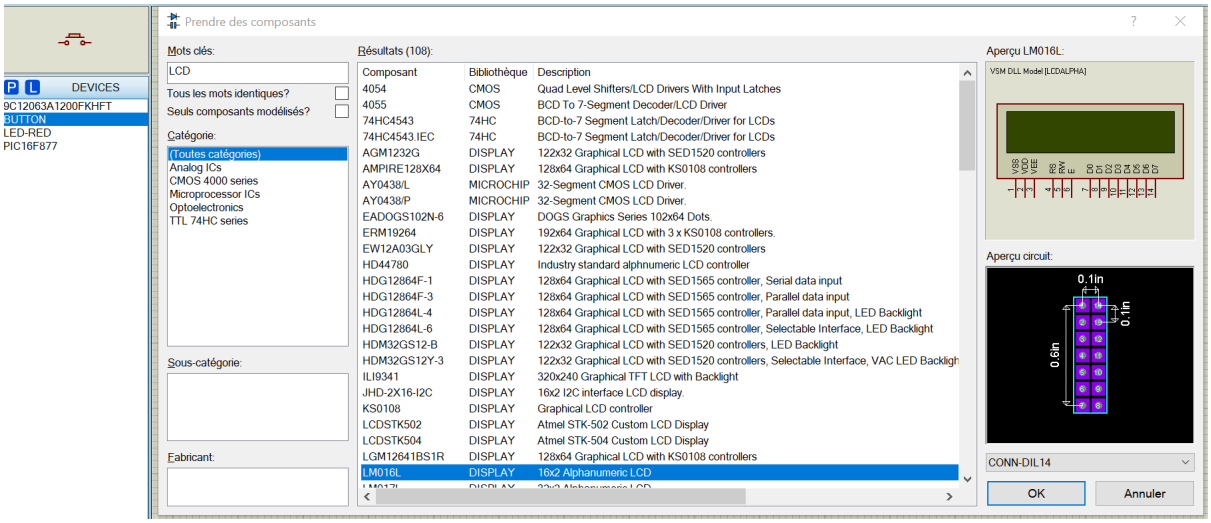
Modifier (afin de satisfaire le cahier des charges) et compiler le projet LED_BP_LCD en ayant ajouter au préalable la librairie LCD



LED_BP_LCD sur PROTEUS

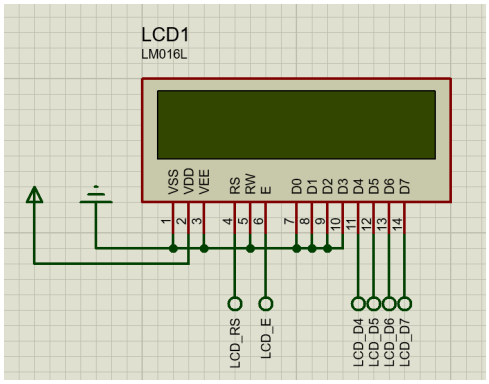
Créer un nouveau projet LED_BP_LCD (en partant du dernier projet LED_BP et enregistrer le projet sous)

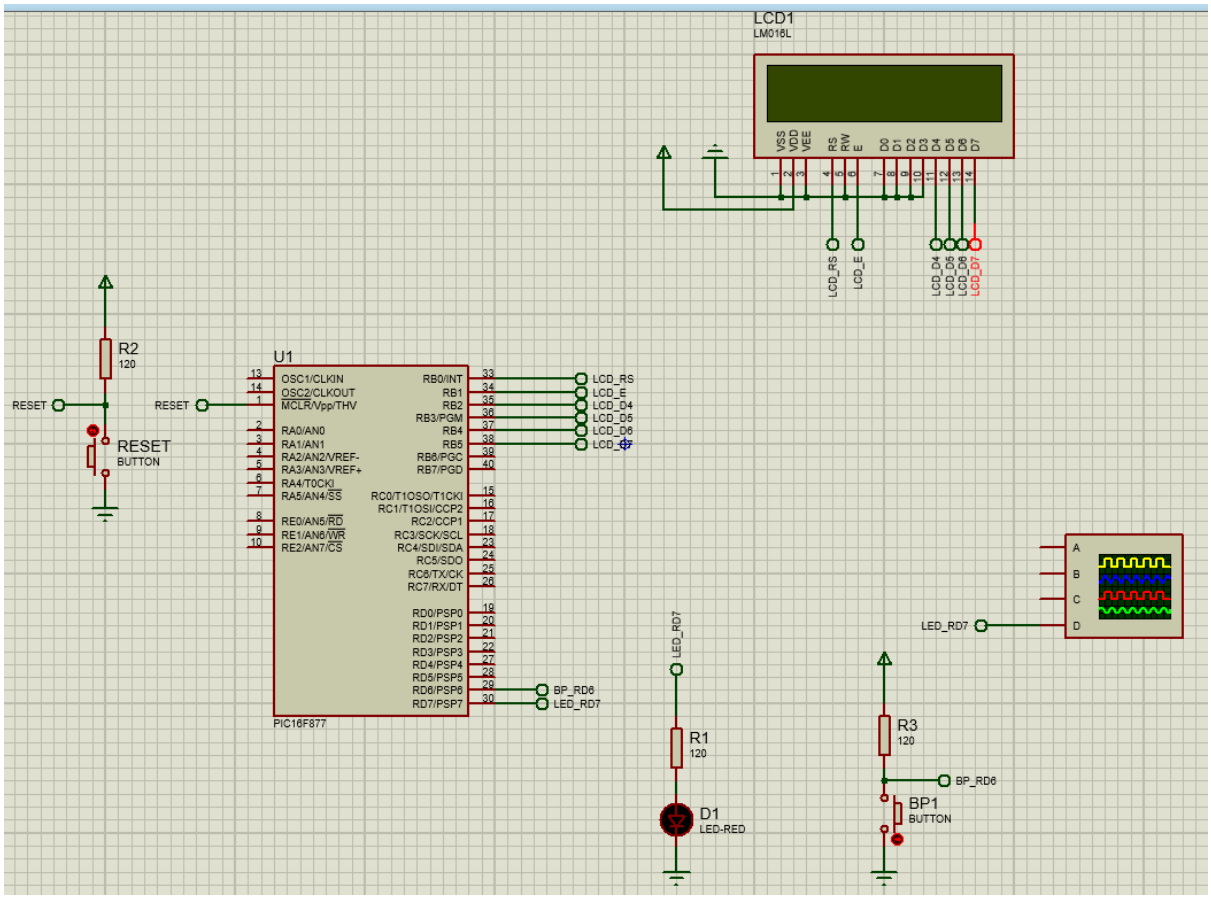
Récupérer un LCD 2×16 le "LM016L" puis ok



Câbler le LCD avec les directives du fichier "LED_BP_LCD.c" de mikroC PRO for PIC.

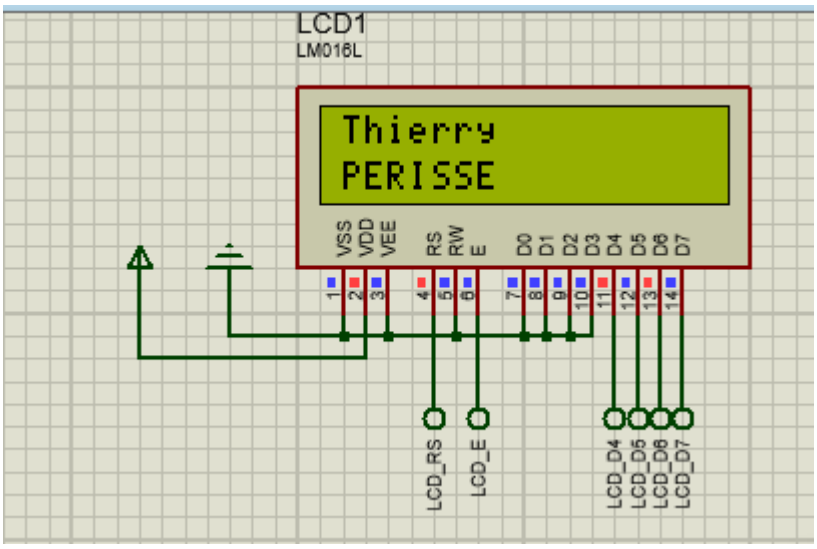
```
/*  Nom du projet : LED_BP_LCD
    Date de test : Juin 2020
    mikroC PRO for PIC v 7.6.0
    @ 8Mhz
    Pour  utiliser  l'afficheur LCD
    on souhaite utiliser le port B avec les
    affectations suivantes : 0,1,5,4,3,2 pour : RS, EN, D7, D6, D5, D4.
    PIC / 16F877
*/
```





Charger le PIC16F877 avec le fichier "LED_BP_LCD.hex"

Lancer la simulation et vérifier le bon fonctionnement du montage



Projet n°4 : LED_BP_LCD_DS1631

Cahier des charges :

Récupérer les informations du capteur I2C DS1631 et afficher la température sur un LCD, si BP OFF Faire clignoter une LED (ON 1s OFF 1s), si BP ON LED OFF. LED sortie 7 du PORTD et BP entrée 6 du PORTD.

LED_BP_LCD_DS1631 sur mikroC PRO for PIC

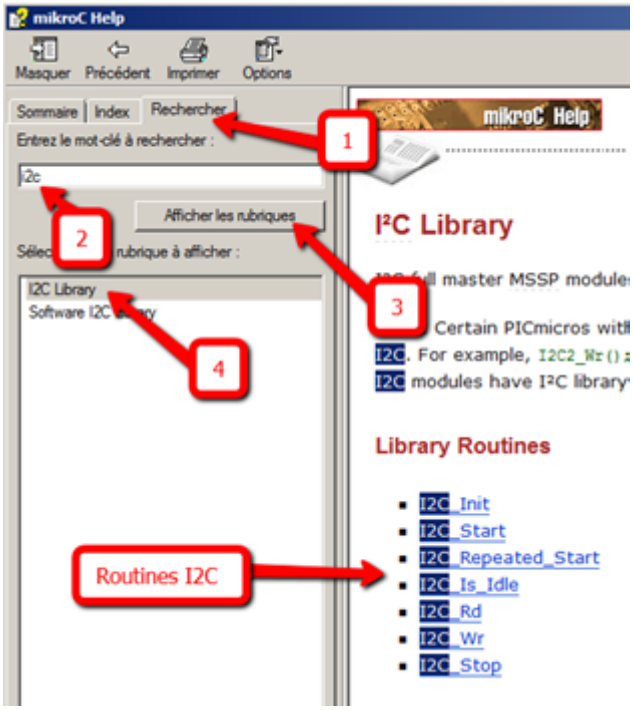
Créer un nouveau projet LED_BP_LCD_DS1631 dans un répertoire ../LED_BP_LCD_DS1631/

Récupérer le fichier "LED_BP_LCD_DS1631.c"

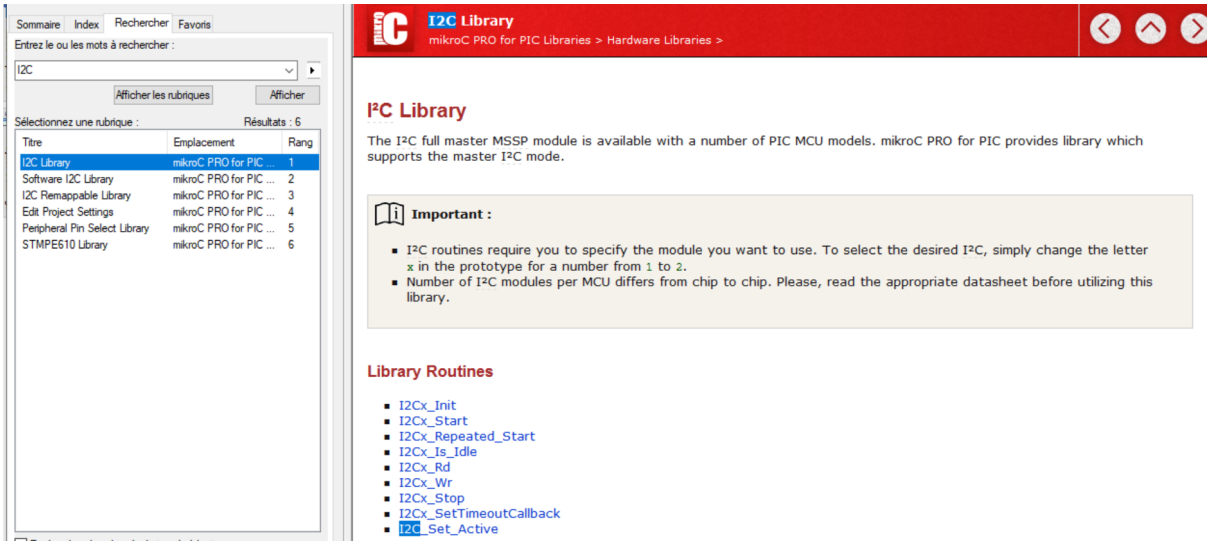
Analyser ce fichier en se servant de la doc constructeur du capteur DS1631.

Regarder les routines I2C (cliquer sur "Help" ou "?" (bleu)

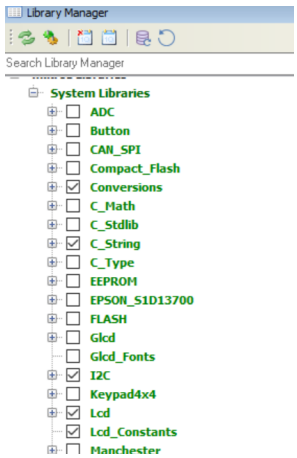
Rechercher, taper "I2C" comme mot à rechercher et valider



Sélectionner soit "I2C Library" ou "Software I2C Library" suivant les routines utilisées :



Dans "library Manager" il faut ajouter les librairies : LCD, I2C, Conversions.

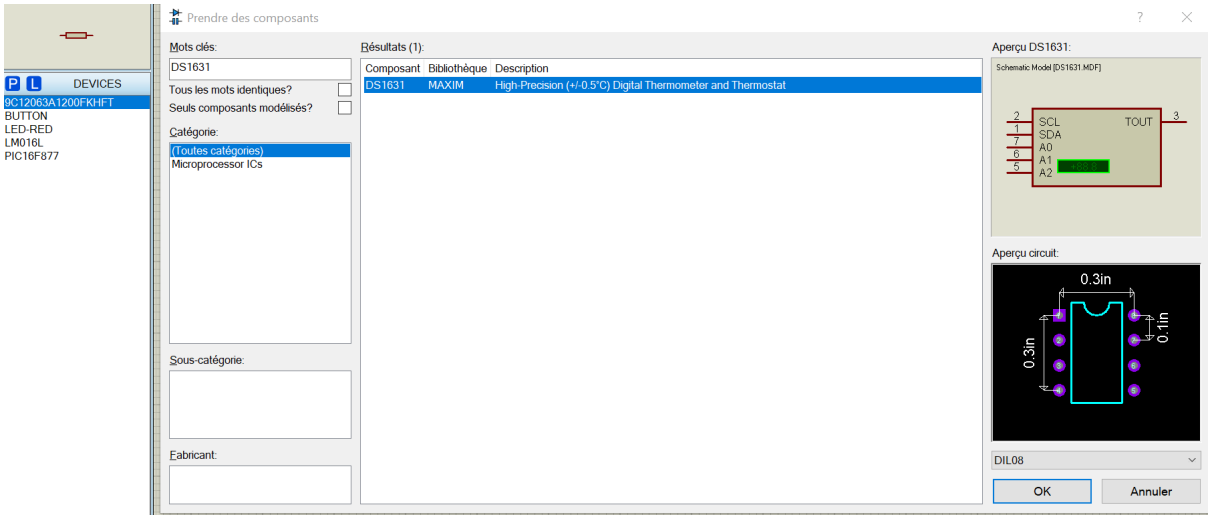


Compiler le projet LED_BP_LCD_DS1631

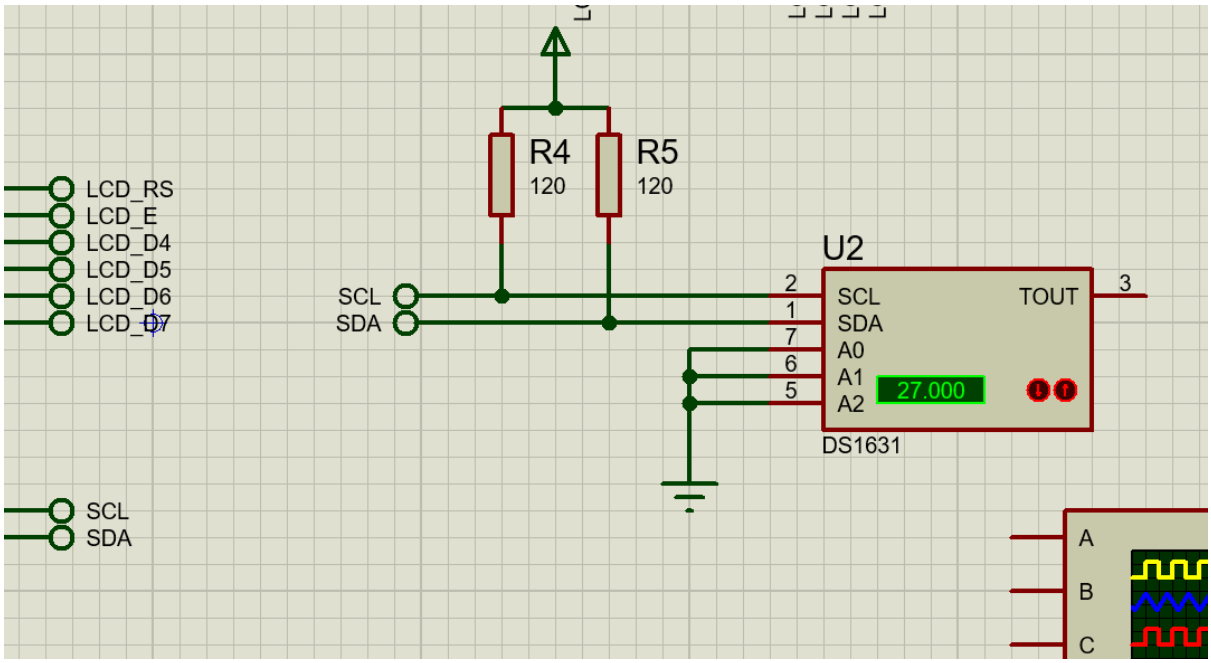
LED_BP_LCD_1631 sur Proteus

Créer le projet LED_BP_LCD_DS1631(en partant du dernier projet LED_BP_LCD et enregistrer le projet sous)

Récupérer le capteur de température I2C le "DS1631" puis ok.



A l'aide de la documentation technique du DS1631, réaliser le câblage avec le microcontrôleur.



Toujours à l'aide de la doc technique :

- Justifier la présence des résistances R4 et R5
- Justifier la mise à la masse de A0, A1, et A2

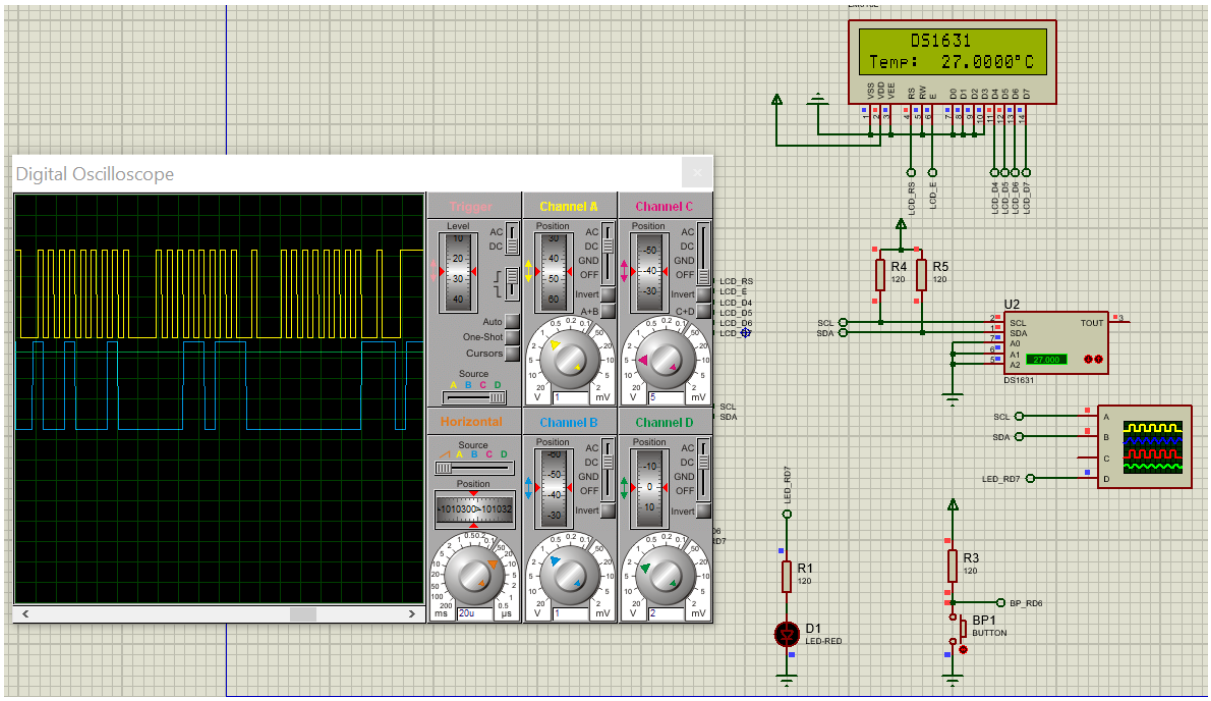
Charger le PIC16F877 avec le fichier "LED_BP_LCD_DS1631.hex"

Lancer la simulation et vérifier le bon fonctionnement du montage :

- Régler une température de 27°C par ex sur le DS1631 et voir si même chose sur le LCD,

Visualiser sur oscilloscope les signaux SDA et SCL :

- Quel est le niveau des signaux SDA et SCL au repos (mesure avec les curseurs de l'oscillo)
- Isoler la condition de start, la condition de stop
- retrouver l'adresse du DS1631 (lecture),
- retrouver les différentes opérations sur les registres, … (en se servant du fichier .c)
- Justifier le contenu de chaque registre
- Visualiser les 2 octets de SDA et SCL donnant la température, calculer la température (cf doc)
- Comparer la température avec SDA et SCL, donner la température réglée sur le DS1631, donner la température affichée sur le LCD.



Projet n°5 : LED_BP_LCD_DS1631_Thermostat

Cahier des charges :

Récupérer les informations du capteur I2C DS1631 et afficher la température sur un LCD, programmer les seuils haut TH et bas TL du Thermostat, valider la sortie Tout du DS1631 suivant le cycle choisi, si BP OFF Faire clignoter une

LED (ON 1s OFF 1s), si BP ON LED OFF. LED sortie 7 du PORTD et BP entrée 6 du PORTD.

LED_BP_LCD_DS1631_Thermostat sur mikroC PRO for PIC

Créer un nouveau projet LED_BP_LCD_DS1631_Thermostat dans le répertoire .../LED_BP_LCD_DS1631_Thermostat

Récupérer le fichier "LED_BP_LCD_DS1631_Thermostat.c"

Ajouter les librairies nécessaires au programme

Compiler le projet LED_BP_LCD_DS1631_Thermostat

LED_BP_LCD_DS1631_Thermostat sur Proteus

Créer un projet LED_BP_LCD_DS1631_Thermostat (avec enregistrer le projet sous du projet précédent)

Pas de câblage supplémentaire

Programmer le PIC16F877 avec le fichier "LED_BP_LCD_DS1631_Thermostat.hex"

Lancer la simulation et vérifier le bon fonctionnement du montage :

Visualiser sur oscillo les signaux SDA et SCL et retrouver les différentes opérations sur le bus I2C :

- Condition de start
- adresse écriture, registres de configurations, adresse lecture, info température, ...

Sur deux températures différentes : (avec partie entière et partie décimale)

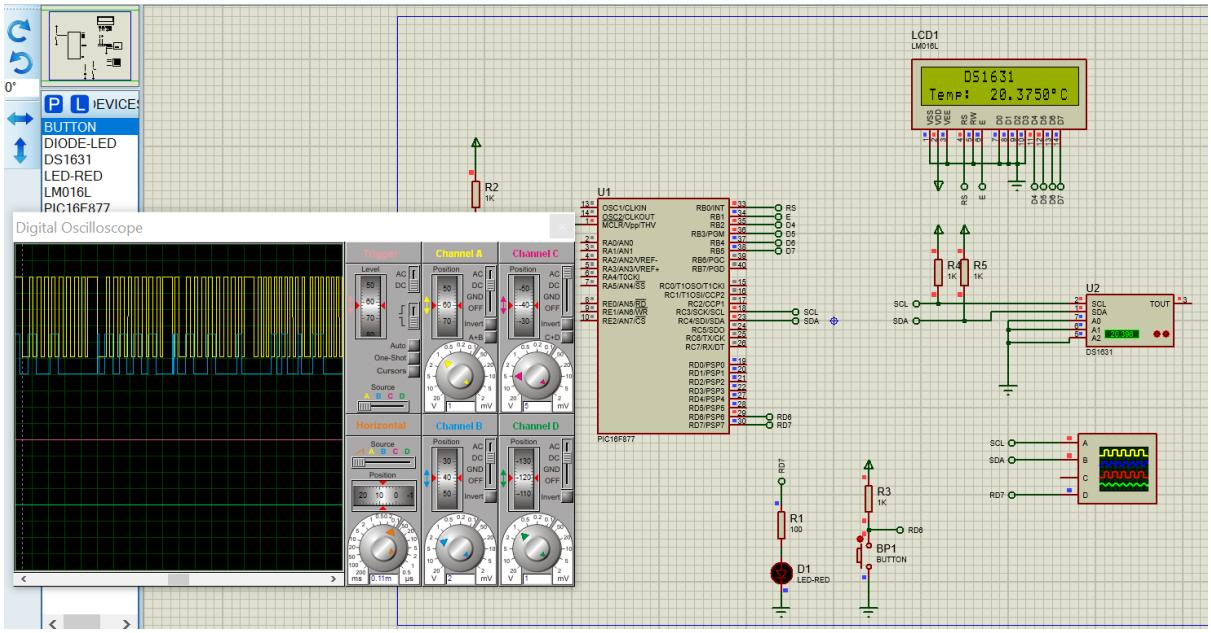
- donner la température avec 2 octets de SDA et SCL,
- donner la température réglée sur le DS1631,
- donner la température affichée sur le LCD.
- Commentaires

Etude du thermostat :

- Tracer Tout en fonction de la température du DS1631 (on peut simuler la variation de la température en jouant sur le + et le - disponible directement

sur le composant)

- En déduire les seuils TH et TL du thermostat.



Projet n°6 : LED_BP_LCD_DS1631_Thermostat_bis

Cahier des charges :

Récupérer les informations (température, TH, TL) du capteur I2C DS1631 et les afficher sur un LCD :

- Si BP OFF : Afficher la température et Faire clignoter une LED (ON 1s OFF 1s),
- Si BP ON : Afficher TH sur la première ligne du LCD et TL sur la deuxième ligne du LCD et LED OFF.

LED_BP_LCD_DS1631_Thermostat_bis sur mikroC PRO for PIC

A faire …

- Envoyer par mail le fichier .c

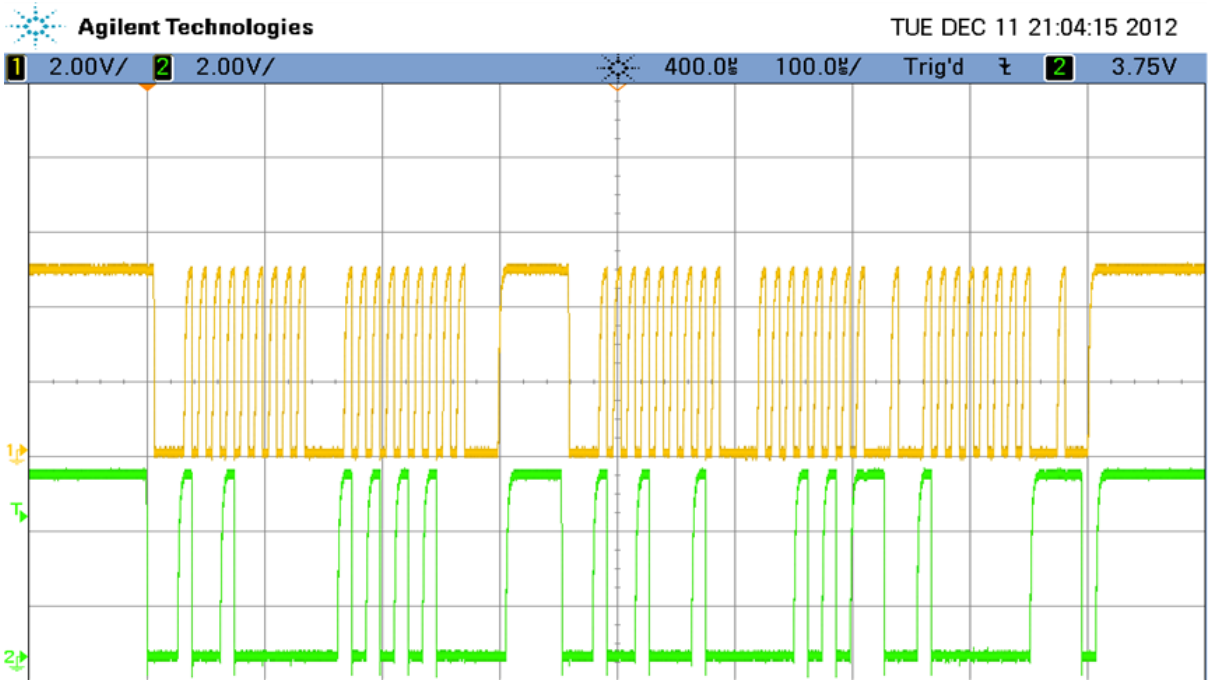
LED_BP_LCD_DS1631_Thermostat_bis sur Proteus

A faire …

- Envoyer par mail le projet PROTEUS.

Préparation : LED_LCD_DS1621 Norme I2C

Ci-dessous un enregistrement des lignes SDA et SCL dans un exemple de communication entre le PIC16F877 et le DS1621 :



LED_LCD_DS1621.c

```
/******Boucle infinie*********************************/
while(1)      {
     I2C_Start();         //Détermine si l'I2C est libre et lance le signal
     I2C_Wr(0x90);        //Mode de contrôle en mode écriture
     I2C_Wr(0xAA);        //Lecture de la température
     I2C_Stop();          //Arrêt du sigal
     I2C_Start();         //Détermine si l'I2C est libre et lance le signal
     I2C_Wr(0x91);        //Mode de contrôle en mode lecture
     MSB = I2C_Rd(1);     //Nombre signé [température entre +125° et -55°C]
     LSB = I2C_Rd(0);     //Si bit 7 = 1 température MSB +0,5°C
     I2C_Stop();          //Arrêt du sigal
     if(MSB<0) {
          MSB = abs(MSB);  // Si temperature négative valeur absolue de MSB
          Temp[0]=45;    }  //Signe - affecté en ascii
     else       {
          Temp[0]=43;  }      //signe + affecté en ascii
     Temp[1]=(MSB/10)+48;         //Valeur des dizaines affecté en ascii
     Temp[2]=(MSB%10)+48;         //Valeur des unités affecté en ascii
     Temp[3]=44;                  //Virgule affecté en ascii
     if (LSB==128) {              //Si bit 7 à 1 alors +0.5°
           Temp[4]='5';  }
     else  {                      //Si bit 7 à 0 alors +0.0°
           Temp[4]='0'; }
     Temp[5]=223;                 // degré affecté en ascii
     Temp[6]=67;                  //C affecté en ascii
lcd(1, 1, 11, temptxt);//Affiche la chaine de caractère temptxt sur le LCD
delay_ms(100);
lcd(2, 1, 7, temp);    //Affiche la chaine de caractère temp sur le LCD
delay_ms(100);
PORTD = ~PORTD;       // toggle PORTD
delay_ms(1000);
     }
```

A l'aide de la boucle infinie ci-jointe et des signaux SDA et SCL ci-dessus :
- Repérer les différents octets présent sur la ligne SDA et en donner leurs significations ?
- Sur cet exemple quelle est la température présente sur le capteur DS1621 ?