


Système Numérique	PIC 16F887 Programmation en langage C et Simulation sous proteus	BTS SN1
		

## 1 Création d'un projet

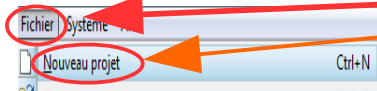
### 1.1 Saisie du schéma

Dans un premier temps nous allons créer un projet Proteus puis un schéma.

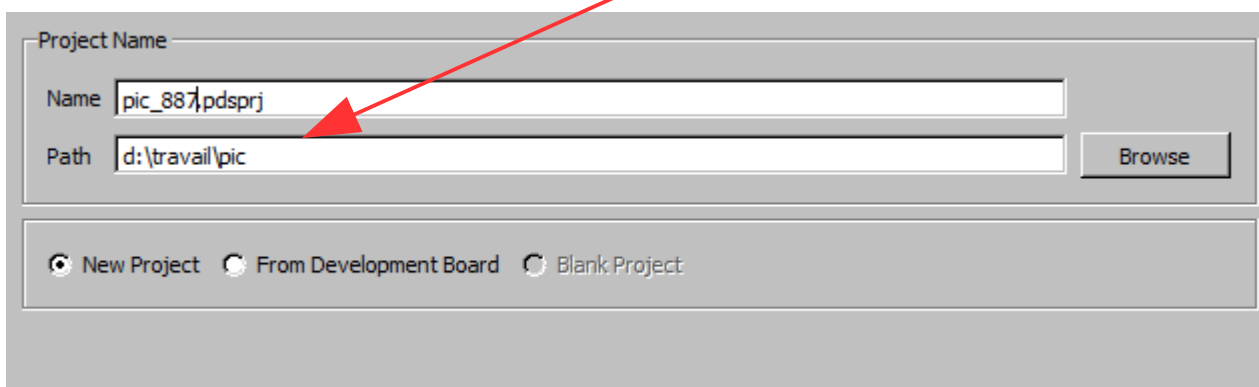
- Lancez le logiciel Proteus 8 .



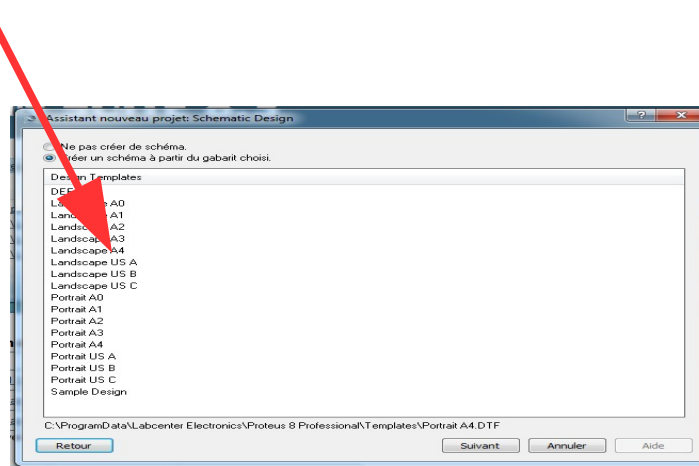
- Créez un projet en choisissant fichier et nouveau projet.



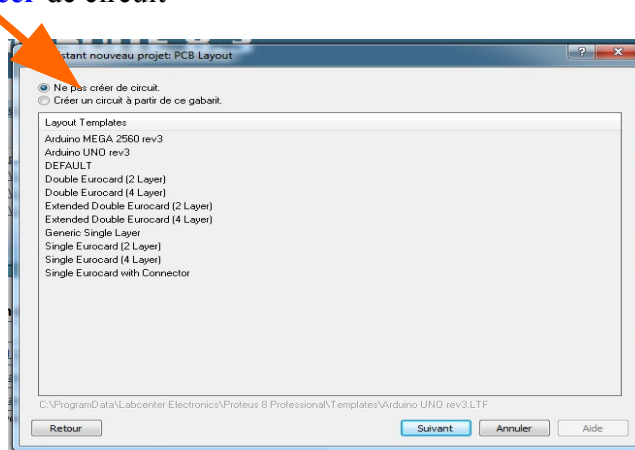
- Placez votre projet dans votre répertoire de travail situé sur le **disque dur de votre ordinateur** et non sur le réseau !!  
Pas d'accents et d'espaces dans vos noms de fichiers et répertoires et pas de caractères spéciaux !!!



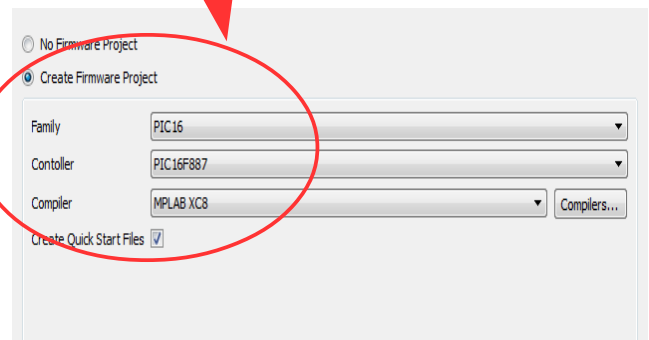
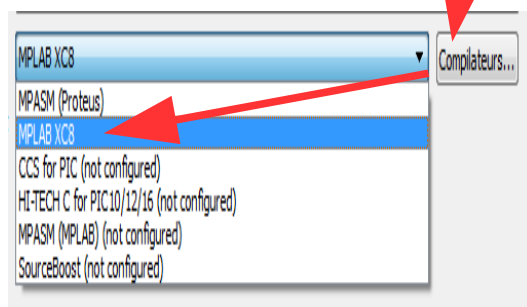
- Choisissez votre gabarit pour votre schéma (portrait A4 pour un schéma simple qui tient sur une page A4)



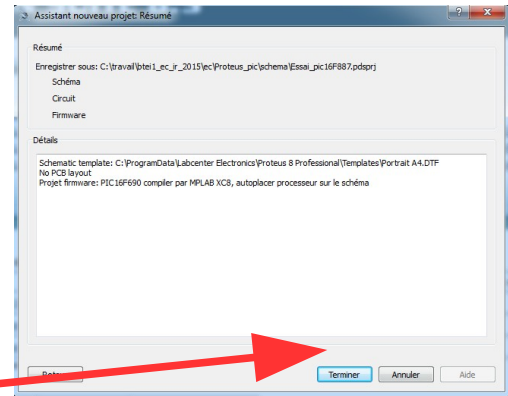
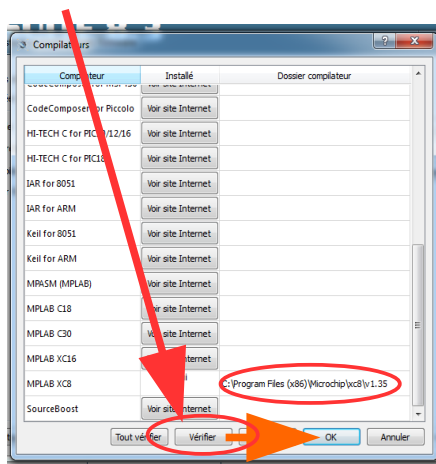
- **Ne pas créer** de circuit



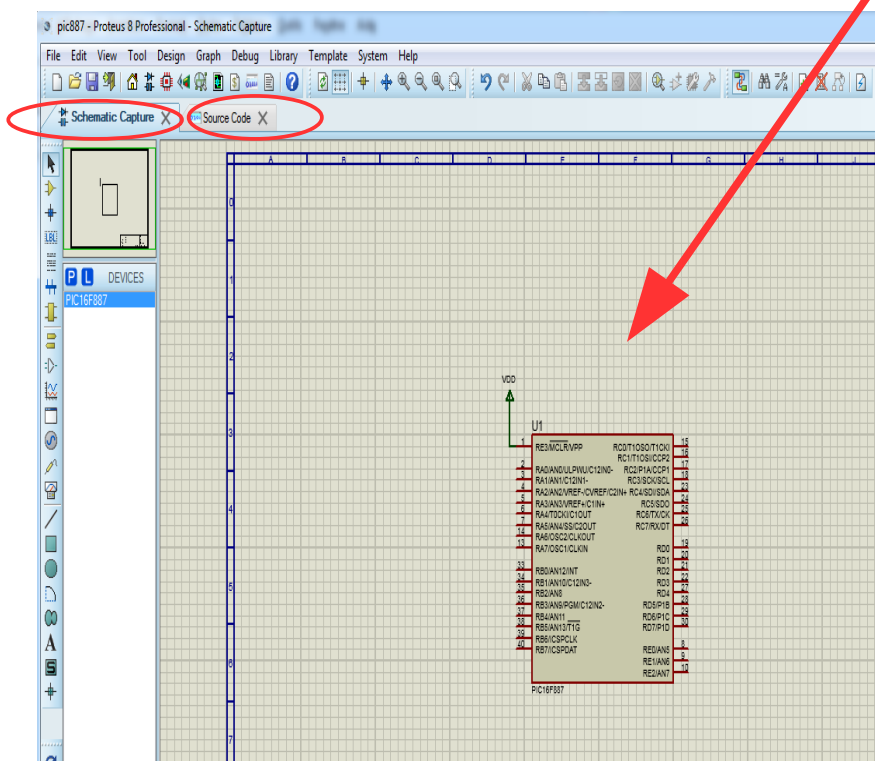
- **Complétez** en fonction de votre **compilateur** ( ici le compilateur de **MPLAB XC8** ) et le **microcontrôleur** choisi (ici un **PIC16F887**).



- **Vérifiez** si le compilateur est bien installé (validez par **OK**)



- **Validez** la fenêtre ci-dessous par «Terminer»
- Vous devriez obtenir la fenêtre suivante qui contient un schéma et un code source.



```
/* Main.c file generated by New Project wizard
 *
 * Created:   lun. sept. 10 2018
 * Processor: PIC16F887
 * Compiler:  MPLAB XC8
 */

#include <xc.h>

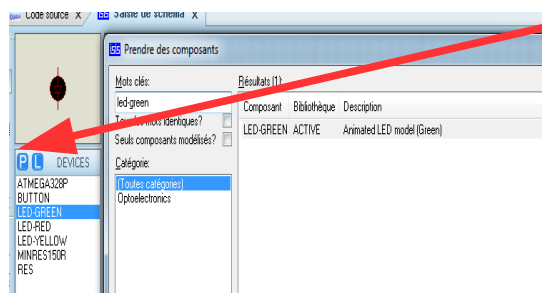
void main(void)
{
    // Write your code here
    while (1);
}
```

- Allez dans l'éditeur de schéma.

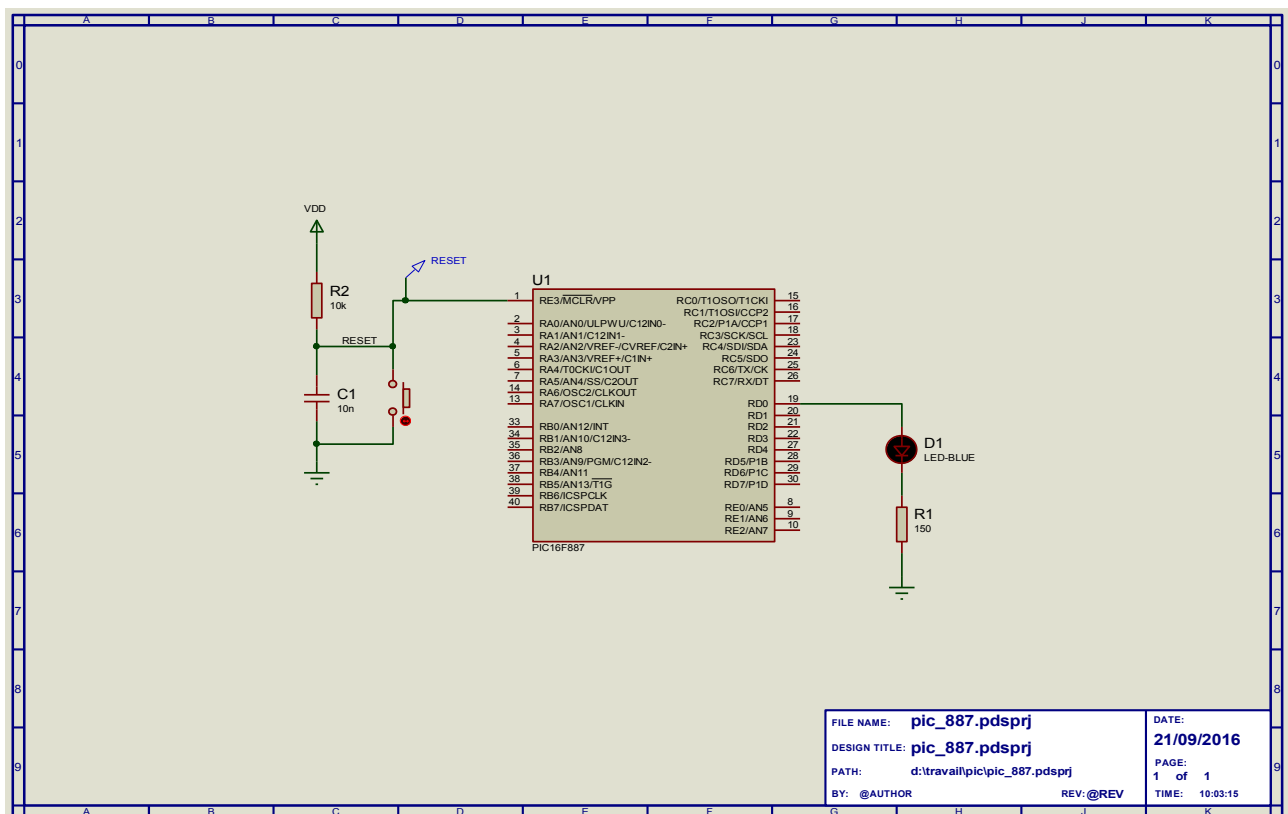


Pour vous aider, utilisez le document **guide\_isis7.1.pdf** situé dans le dossier **Notices/Proteus/**

- **Placez** une led bleue et une résistance de 150  $\Omega$  avec l'outil de placement des composants



- **Placez et Reliez** les composants comme ci-dessous.



## 1.2 Édition du code source

On se propose de faire clignoter la Led D1.

### 1.2.1 Utilisation de l'éditeur intégré

- Ouvrez l'éditeur de code de Proteus.



- Complétez le code ci-dessous

```
#include <xc.h>

// Directives
#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF // Watchdog Timer Disable bit

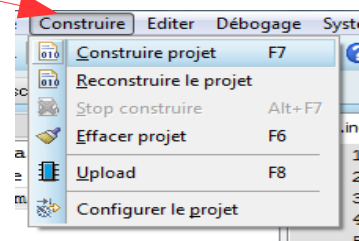
#define _XTAL_FREQ 4000000

// Fonction initialisation des entrées et sorties
void init(void)
{
    TRISD = .....; // A compléter : configuration de la broche RD0 en sortie
    .....
}

void main()
{
    init(); // Appel de la fonction initialisation des entrées sorties
    while(1)
    {
        // A compléter
        ..... // Allumer la led
        __delay_ms(1000); // Temporisation de 1000 ms ( 1 s)
        ..... // Eteindre la led
        __delay_ms(1000); // Temporisation de 1000 ms ( 1 s)
    }
}
```

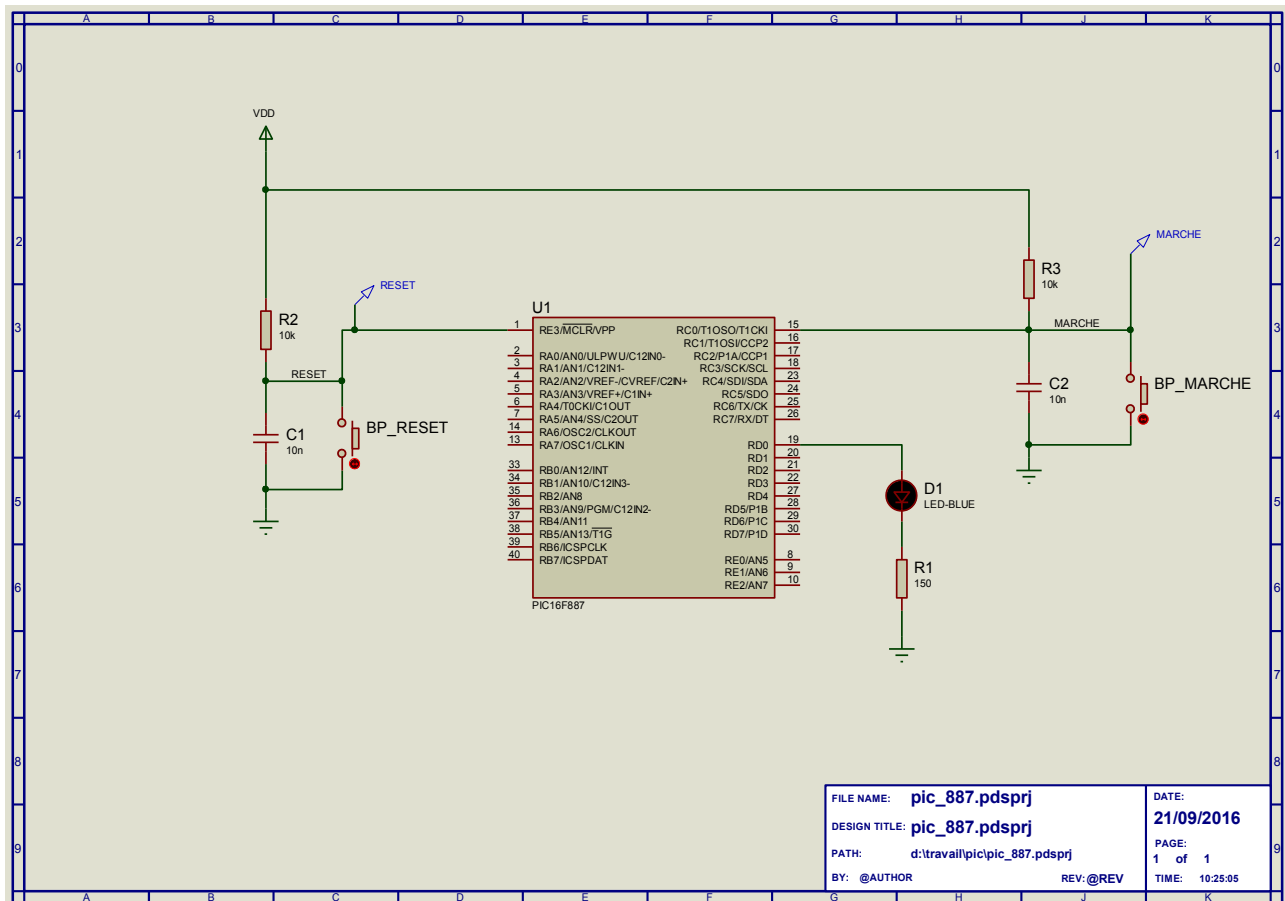
- Après avoir complété votre programme, **compilez** Build et Build Project ( ou CTRL F7), **corrigez** vos éventuelles erreurs .

**Après chaque modification** de votre code, il faut reconstruire le projet.

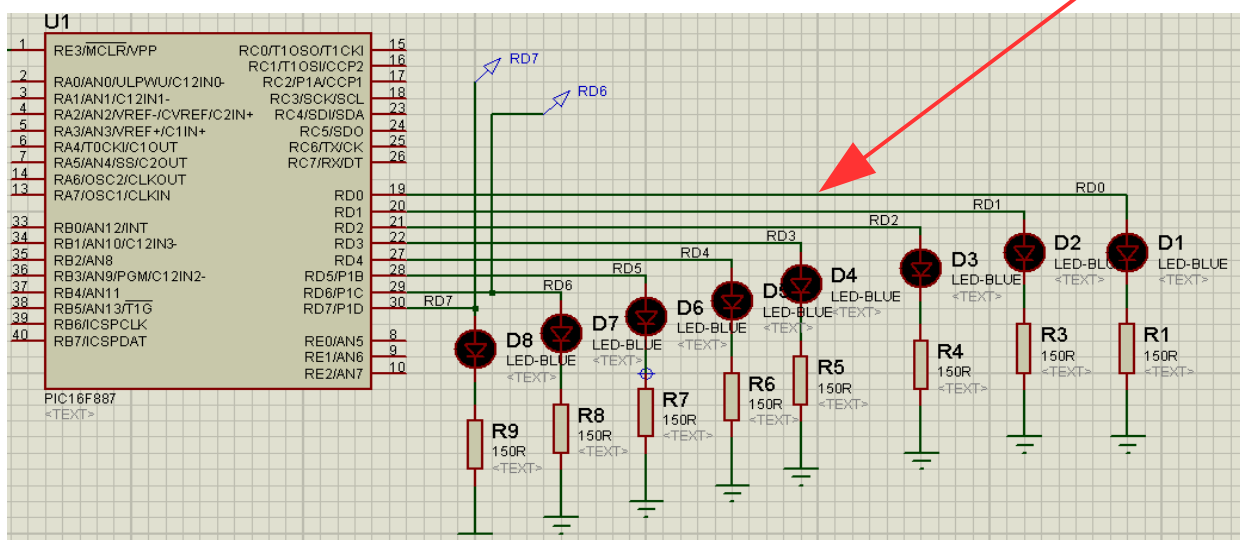


- Lancez la simulation et vérifiez que la led D1 clignote. **corrigez** vos éventuelles erreurs.
- On souhaite faire clignoter la led D1 uniquement quand on appuie sur le bouton poussoir BP\_MARCHE. Compléter le schéma comme ci dessous.
- Dans le code source, configurer la broche RC0 en entrée avec TRISC =.....;
- Modifier le programme précédent pour savoir si le bouton poussoir est pressé et ainsi faire clignoter la led D1.
- **Compilez, corrigez** vos éventuelles erreurs.

- Lancez la simulation et vérifiez que la led D1 clignote quand on actionne le bouton poussoir BP\_MARCHE.



- On souhaite écrire un programme qui exécute un chenillard avec les leds D1 à D8.



- Donner le nom du registre et la valeur à mettre dans ce registre pour configurer en sortie toutes les broches qui permettent de commander les leds D1 à D8.
- Écrire le programme CHENILLARD.C
- Compiler, simuler et vérifier le fonctionnement.
- Programmer le micro-contrôleur avec le logiciel PICKIT et vérifier son bon fonctionnement.
- On souhaite simplifier le programme du chenillard en utilisant les opérateurs de décalage << ou >>. Modifier le programme précédent et tester le fonctionnement.