****

**FORMATION SUR**

**Maquette de développement PIC16FXXX - LS-PIC1**

****

Sommaire

[**I-** **Présentation et caractéristiques techniques** 3](#_Toc536828035)

[**1-** **Présentation :** 3](#_Toc536828036)

[**2-** **Caractéristiques techniques** 3](#_Toc536828037)

[3- **Les modules principaux** 4](#_Toc536828038)

[**3-1-** **Schéma électrique de carte développement ( 18 broches ) :** 4](#_Toc536828039)

[**3-2-** **Schéma électrique de carte développement ( 28 broches : 16F876 ) :** 4](#_Toc536828040)

[**3-3-** **Schéma électrique de carte développement ( 40 broches : 16F877 ) :** 5](#_Toc536828041)

[**3-4-** **Interface des entrées TOR :** 5](#_Toc536828042)

[**3-5-** **Interface des sorties LED :** 6](#_Toc536828043)

[**3-6-** **Interface de puissance avec relais** 6](#_Toc536828044)

[**3-7-** **Interface d’isolation galvanique (opto-coupleur TLP521-4 )** 7](#_Toc536828045)

[**3-8-** **Carte LCD 1602** 8](#_Toc536828046)

[**3-9-** **Carte afficheur 7 seg** 8](#_Toc536828047)

[**3-10-** **Clavier matriciel 4 x 4** 8](#_Toc536828048)

[**3-11-** **Carte d’interfaçage RS232-USB (TX et RX)** 9](#_Toc536828049)

[**II-** **Environnement de programmation :** 10](#_Toc536828050)

[1- Etapes de programmation : 10](#_Toc536828051)

[Activité n° 1: Découverte de l’environnement programmation textuelle 11](#_Toc536828052)

[Activité n° 2: Compteur Modulo 10 15](#_Toc536828053)

[Activité n° 3: Gestion d’un afficheur LCD 17](#_Toc536828054)

[Activité n° 4: CAN - Voltmètre 18](#_Toc536828055)

[Activité n° 5: MLI – CONVERSIO N 19](#_Toc536828056)

[Activité n° 6: Interface UART + LABVIEW 20](#_Toc536828057)

1. **Présentation et caractéristiques techniques**
2. **Présentation :**

La valise **LS-PIC1** est une maquette de développement PIC 16Fxxx permettant l’étude sur des différentes type de microcontrôleur PIC (microchip ) forme PDIP :

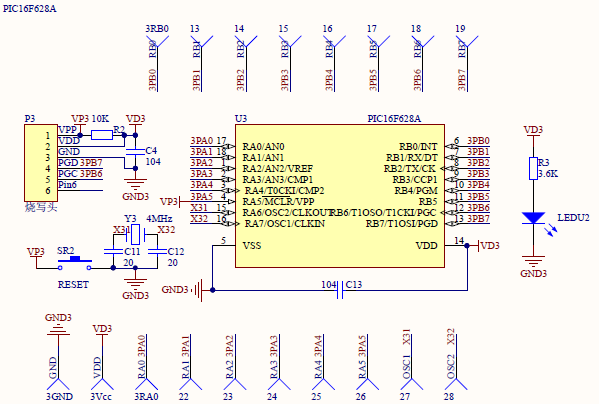
1. PIC de 18 broches ( 16F84, 16F628, 16F88….)
2. PIC de 28 broches ( 16F876..)
3. PIC de 40 broches (16f877 …)

Elle est principalement composée de :

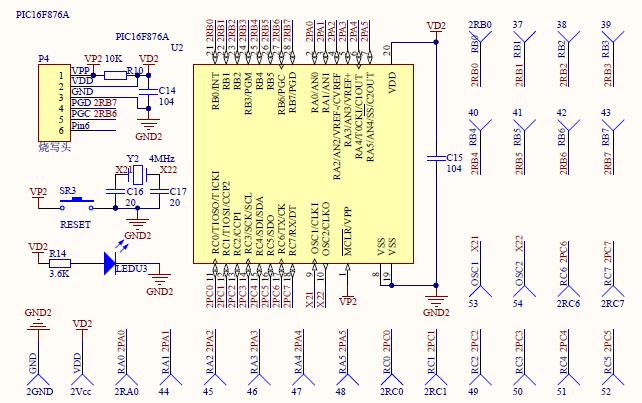
1. Trois cartes de développement (PIC 18, 28 and 40 broches)
2. Programmateur **Microchip PicKit 2**
3. Deux Interface entrées TOR (12 interrupteurs et 16 boutons poussoirs : 12 NO et 4 NC )
4. Interface sorties LED
5. Interface de puissance avec relais
6. Interface opto-coupleurs
7. Carte LCD 1602
8. Carte afficheur 7 seg
9. Carte d’entrée analogique (utilisation de module CAN)
10. Clavier matriciel 4 x 4
11. Carte d’interfaçage RS232-USB (TX et RX)
12. Plaque d’essai large
13. **Caractéristiques techniques**

* Tension d’alimentation : 220ACV±5% 50Hz
* Tension d’alimentation des circuits : +5VDC
* Puissance de l’ensemble : 5W
* Environnement : température -10℃～+60℃ Humidité＜90％(25℃)

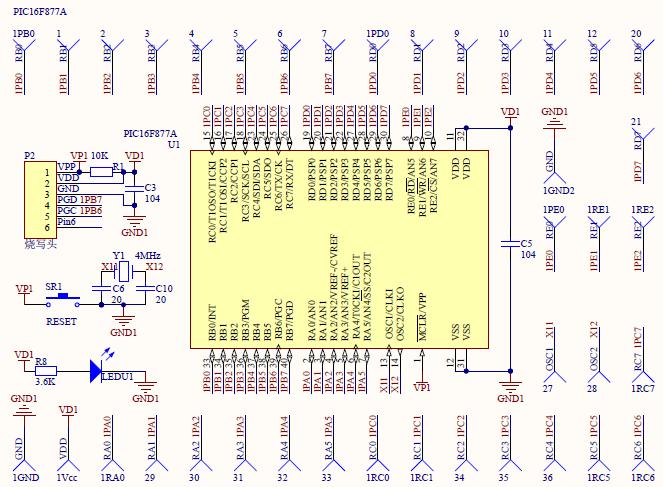
1. **Les modules principaux** 
   1. **Schéma électrique de carte développement ( 18 broches ) :**



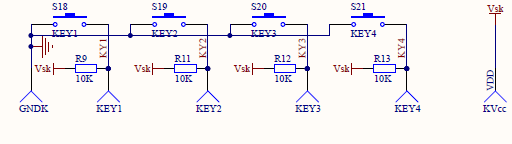
* 1. **Schéma électrique de carte développement ( 28 broches : 16F876 ) :**

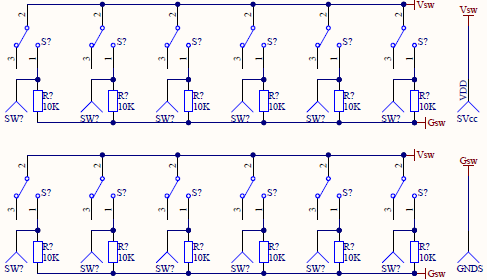


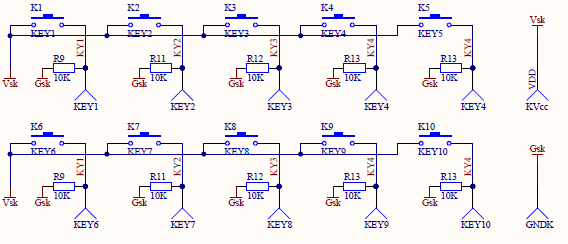
* 1. **Schéma électrique de carte développement ( 40 broches : 16F877 ) :**



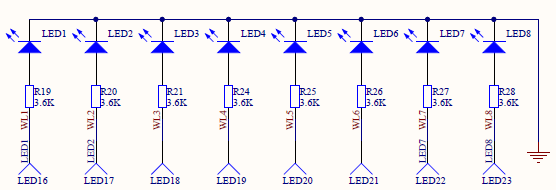
* 1. **Interface des entrées TOR :**

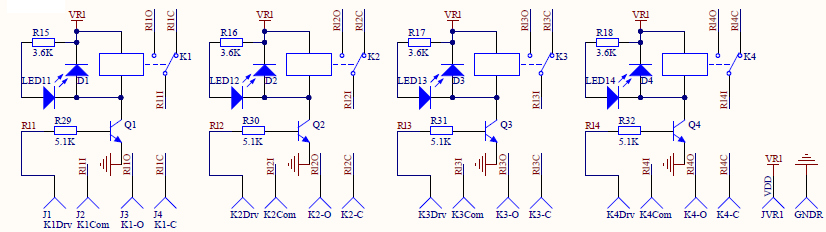






* 1. **Interface des sorties LED :**



* 1. **Interface de puissance avec relais**

**Description: Relais 5v-220 10A**

Un relais électronique est un interrupteur qui se commande avec une tension continue de faible puissance. La partie interrupteur sert à piloter des charges secteur de forte puissance (jusqu’à 10A couramment).

**Fiche technique Model: SRD-5VDC-SL-C :**

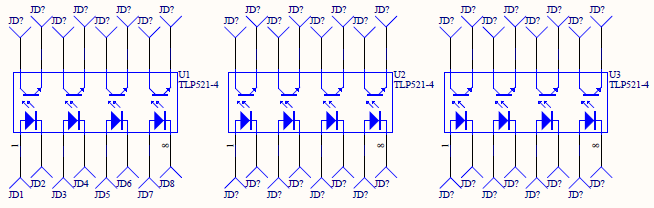
1. Coil voltage: DC 5V
2. Coil resistance: 70Ω ~ 80Ω
3. Contact resistance: 100Ω Max
4. Operation time: 10msec Max
5. Release time: 5msec Max
6. Operating current: 43mA ~ 46mA
7. Release current: 15mA ~ 18mA
8. Pin: 5Pin
9. Insulation resistance: ≥100M (Ohm)
10. Between coil & contacts: AC 1500V 50HZ ~ 60HZ/min
11. Between contacts: AC 1000V 50HZ ~ 60HZ/min
12. Rated load:: 10A 250V AC / 10A 125V AC / 10A 30V DC / 10A 28V DC

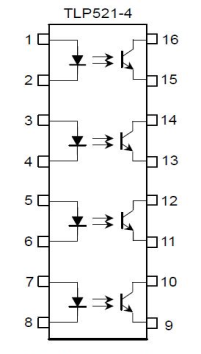
**Branchement du relais**

## RÃ©sultat de recherche d'images pour "relais 5v"

## 

* 1. **Interface d’isolation galvanique (opto-coupleur TLP521-4 )**



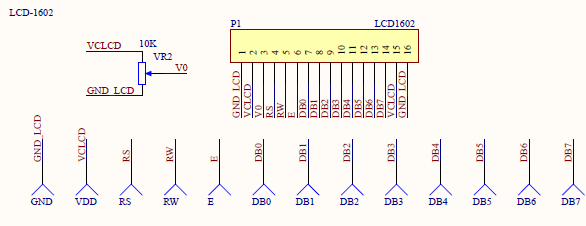


**Le TLP521-4** est un isolateur traversant couplé optiquement dans un boîtier DIP 16 broches L'isolateur se composent de diodes émettrices de lumière infrarouge et de photo transistor NPN silicium. Cet isolateur est largement utilisé pour les terminaux informatiques, les contrôleurs de système industriel, la transmission du signal entre les systèmes de différents potentiels et impédances.

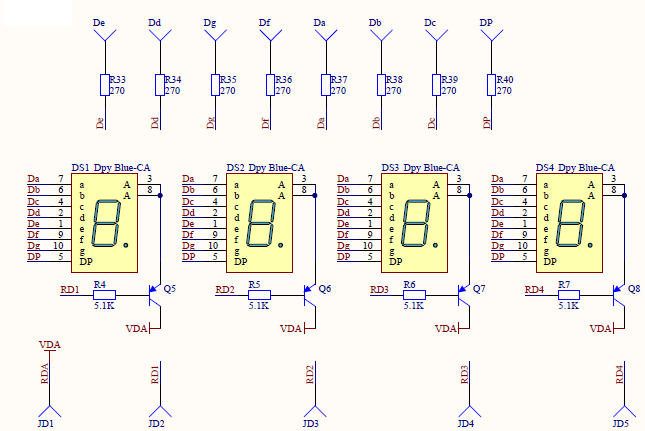
* Un ratio élevé de transfert de courant de 50%
* Haute tension d'isolement de 7,5K
* BVCEO élevé de 55V(min)
* Reconnu UL E91231
* Sélections personnalisée électriques disponibles
* Courant If (Forward) de 50mA
* Quad canaux

**Applications**

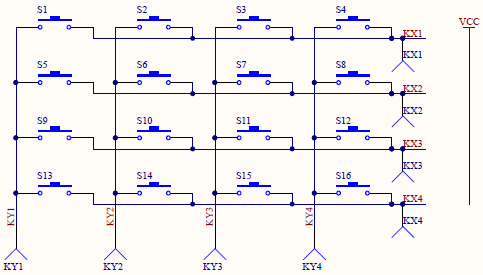
* Industrie, Test et Mesure, Traitement du Signal
  1. **Carte LCD 1602**

****

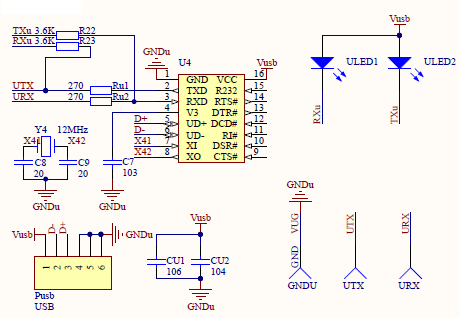
* 1. **Carte afficheur 7 seg**



* 1. **Clavier matriciel 4 x 4**



* 1. **Carte d’interfaçage RS232-USB (TX et RX)**



1. **Environnement de programmation :**
2. Etape de programmation :

**Cahier de charge**

**Algorithme**

**Programme**

**Traduction de l’algorithme en un programme en language évolué (Pascal )**

**En utilisant compilateur MIKRPASCAL**



**Compilation**

**Transformation du programme en language machine et génération du ficher (.hex)**

**Chargement**

**Charger le ficher (.hex) dans la mémoire de microcontrôleur en utilisant logiciel PICkit 2**

****

**Mise en oeuvre**

**Mise en marche et exploitation du système à base de microcontrôleur**

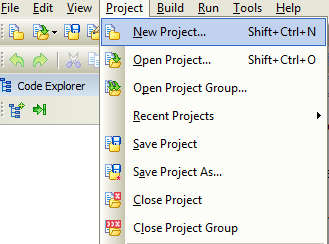


**Simulation**

**Simuler le fonctionnement sur PC à l’aide d’un logiciel de simulation (ISIS..)**



## Activité n° 1: Découverte de l’environnement de programmation textuelle



**1**

Lancer le logiciel mikropascal pro et créer un nouveau projet :

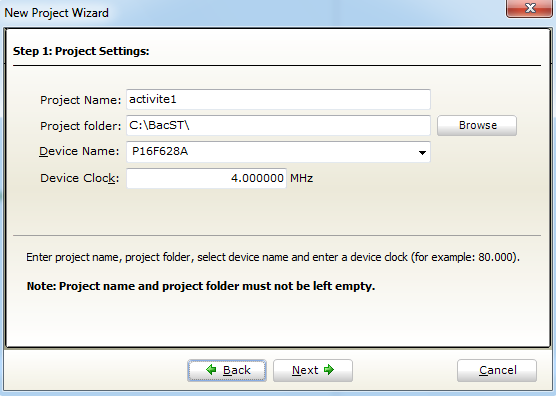
1. Dans le menu principal cliquer sur « ***Project****»* par la suite sur « ***New project****»*



**2**

1. Une fenêtre « *New Project Wizard* » apparait, cliquer sur « ***Next***»
2. Dans le champ « ***Project Name*** » Saisir le nom du projet

Par exemple : activite1, ce nom ne doit pas contenir des caractères accentués ou des espaces.



1. Cliquer sur le bouton «**Browse** » et sélectionner un répertoire sur le disque de votre PC, ce répertoire contiendra tous les fichiers de votre projet.

**4**

**3**

1. Sélectionner le type du microcontrôleur : **P16F628A**

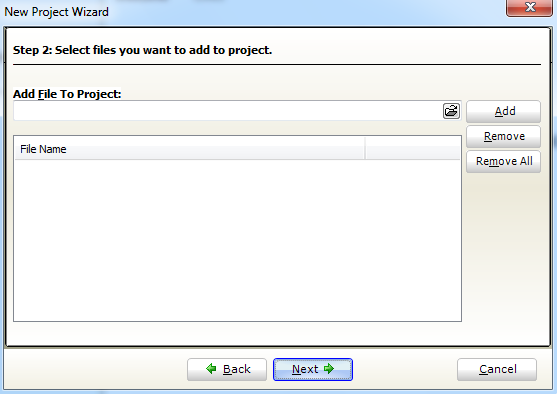
**12.000000**

**6**

**5**

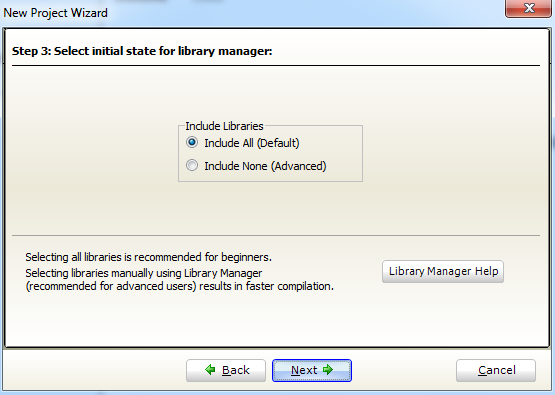
1. Fixer la valeur de l’horloge du microcontrôleur : **4Mhz**.
2. Cliquer sur le bouton « **Next** »

**7**



**8**

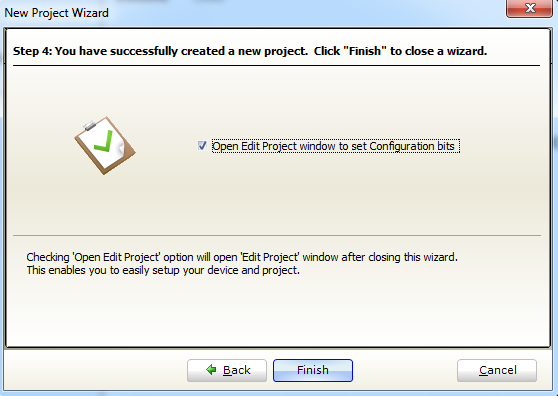
1. Cliquer sur le bouton « **Next** »



**9**

**10**

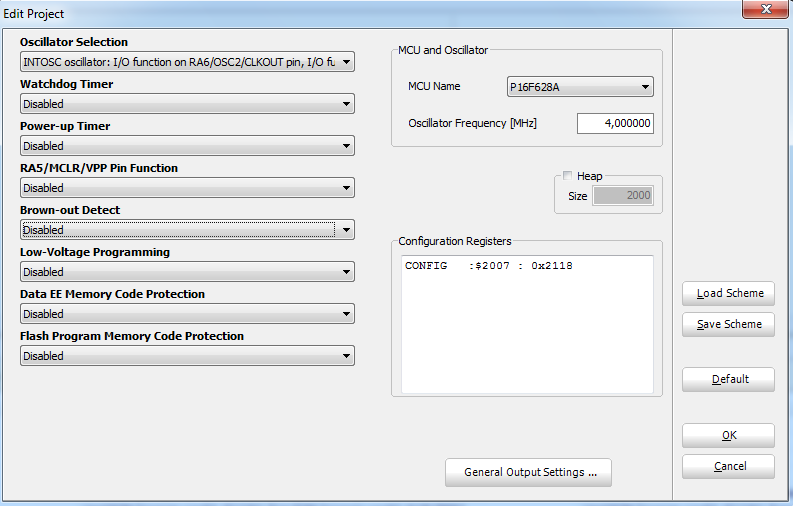
1. Cocher la case « **Include All**» cette option permet d’utiliser les fonctions prédéfinies de Mikropascal
2. Cliquer sur le bouton « **Next** »



**11**

**12**

1. Cocher la case « ***Open Edit Project window to set Configuration bits*** »
2. Cliquer sur le bouton « ***Finish****»*

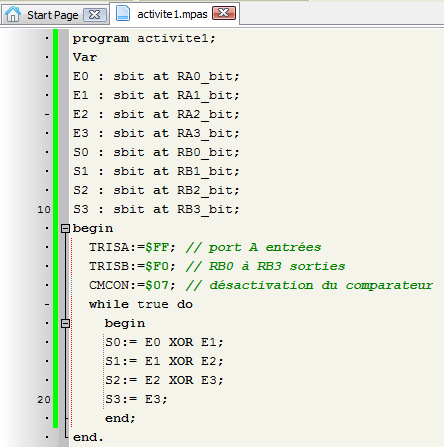


**12.000000**

**13**

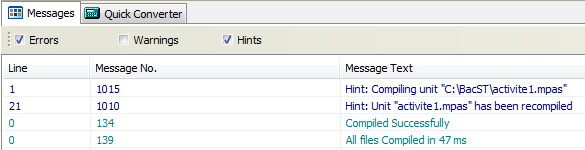
**14**

1. **Assurer les réglages des bits de configurations** comme indiqué dans la figure précédente
2. Cliquer sur le bouton « **OK** »



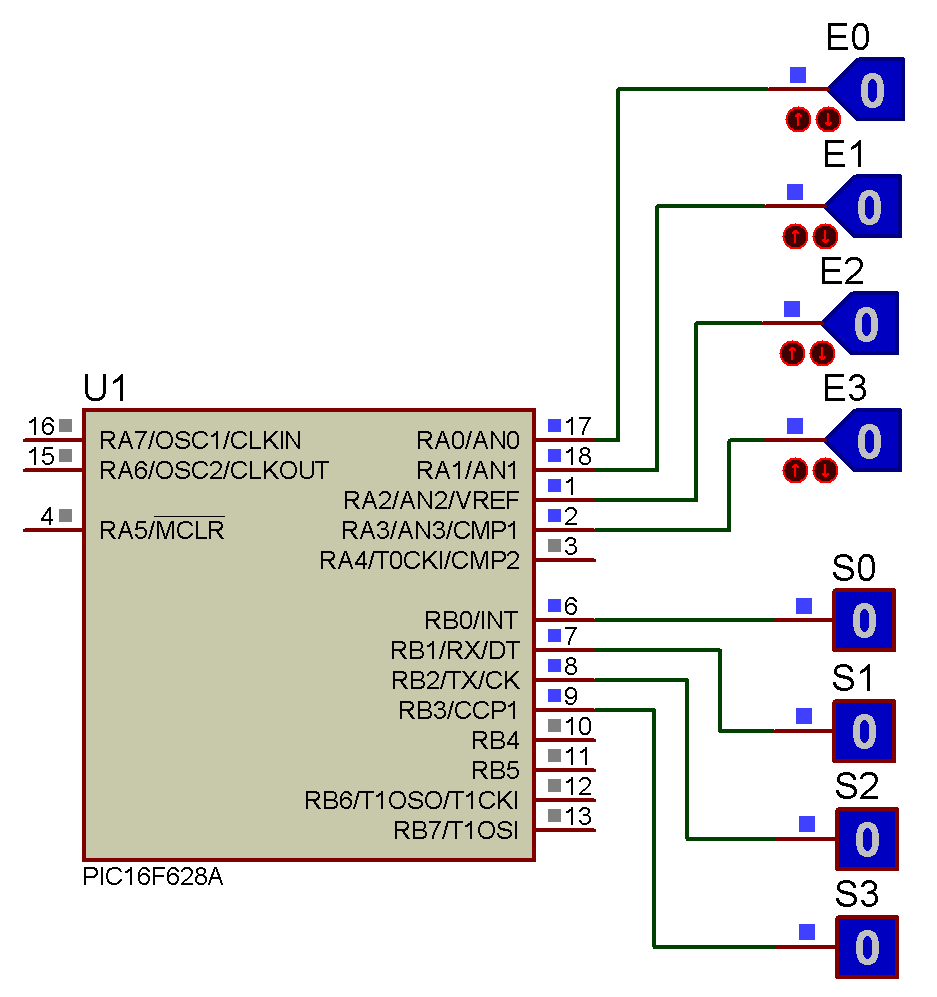
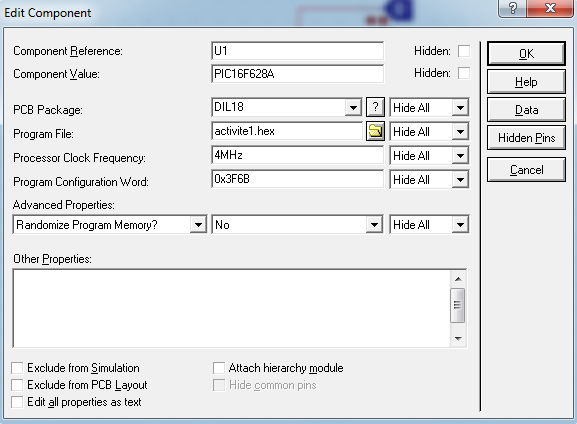
**156**

1. Saisir le programme ci-contre
2. Compiler le projet et vérifiez que la compilation a réussi : message «*Compiled Successfuly*» dans l’onglet messages.



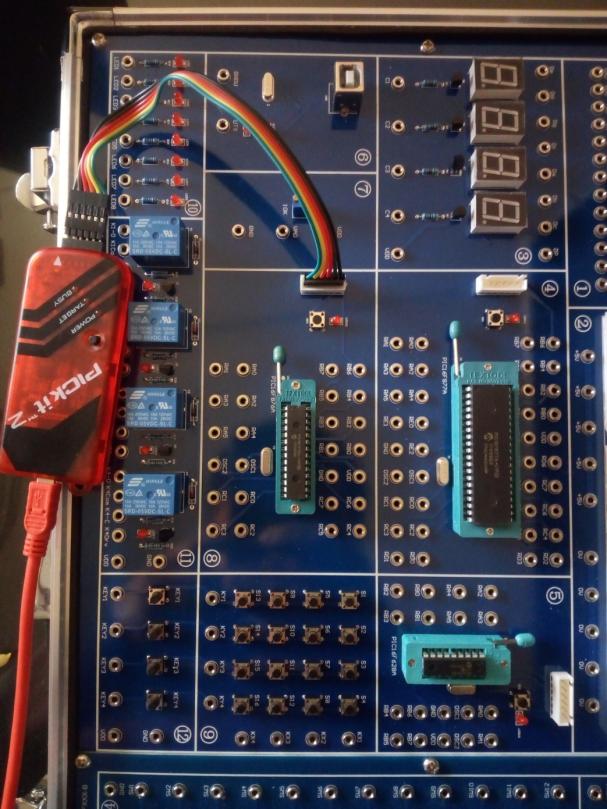
**166**

**166**

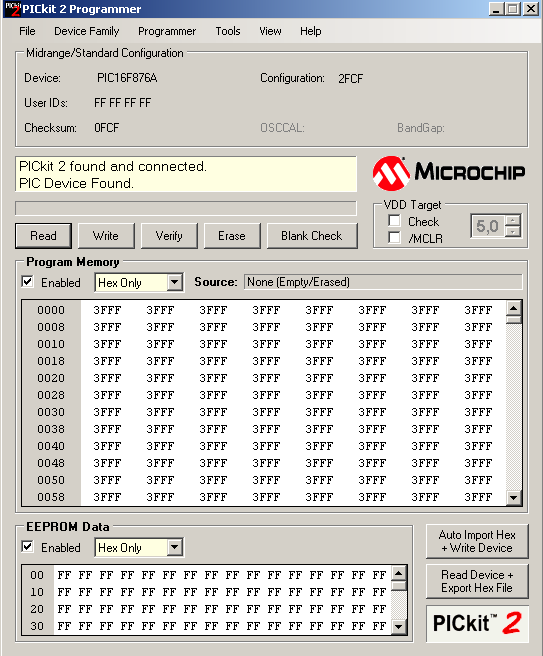
1. Lancez le logiciel « ISIS » et saisissez le schéma de simulation.
2.  Dans la fenêtre « Edit Component » indiquer le nom et le chemin du fichier .hex
3.  Régler l’horloge à **12MHz**

**18**

**19**



1. Bancher le programmateur PICKIT 2 au PC via le câble USB et connecteur de programmation de la valise
2. Lancer le logiciel PICkit 2.
3. Cliquer sur l’icône de Read pour détecter le type du microcontrôleur.
4. Dans le menu principal cliquer sur : *File… import … fichier.hex*
5. Programmer le microcontrôleur PIC16F876A



**23**

**24**

**22**

**Remarque :** le câble ICSP doit être connecté au PICkit2 et connecteur de programmation de microcontrôleur de cette manière :

Le fil noir au indicateur de coté PICkit2 de l’autre de partie de connecteur de programmation prés de la diode rouge





**Attention il ne faut surtout pas avoir à la fois une alimentation par PICkit 2**

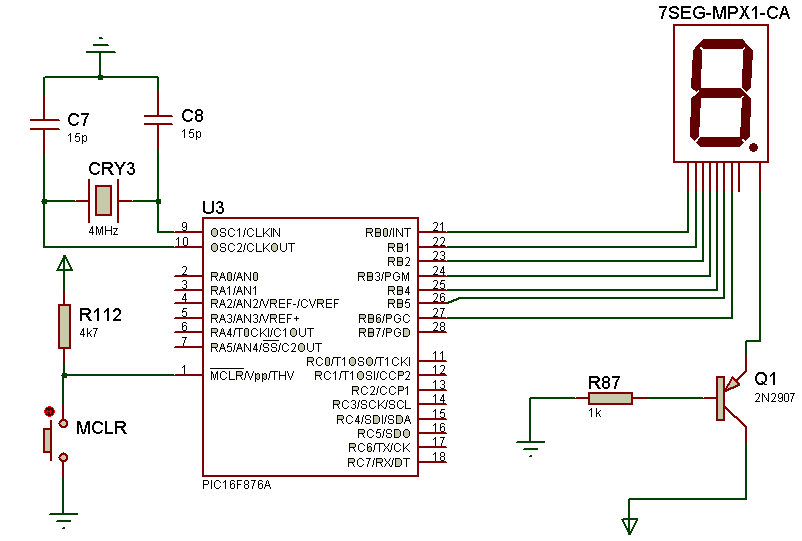
**et une alimentation externe : RISQUE DE COURT CIRCUIT ! Et**

**DESTRUCTION de PICKIT !**

# 

## **Activité n° 2: Compteur Modulo 10**

On souhaite réaliser un compteur modulo 10 en utilisant la maquette de développement PIC16FXXX conformément au montage suivant :



**C1 vers 0V**

**Les segments [Da Db Dc Dd De Df Dg]**

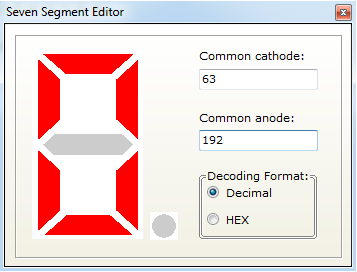
**Vers RB0 ……RB6**

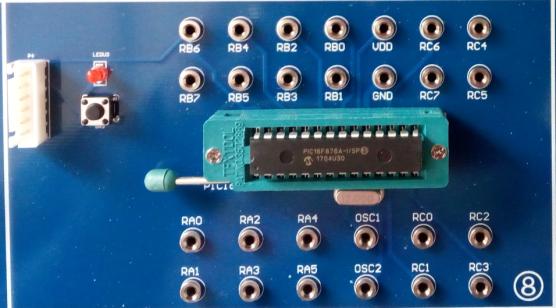
**VDD du bloc afficheur 7**

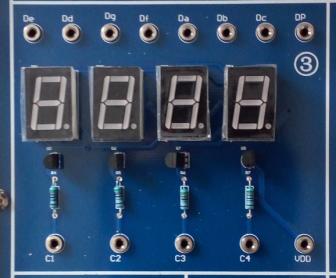
**vers**

**+5V du bloc alimentation**

1. Reliez **C1** du bloc afficheur 7 segments à **0V** du bloc alimentation
2. Reliez **VDD** du bloc afficheur 7 segments à **5V** du bloc alimentation
3. Reliez les entrées **des segments [Da Db Dc Dd De Df Dg]** respectivement aux broches **RB0, RB1, RB2, RB3, RB4, RB5 et RB6** du microcontrôleur PIC16F876A.
4. Essayer d’utiliser le gestionnaire d’afficheur 7 segment





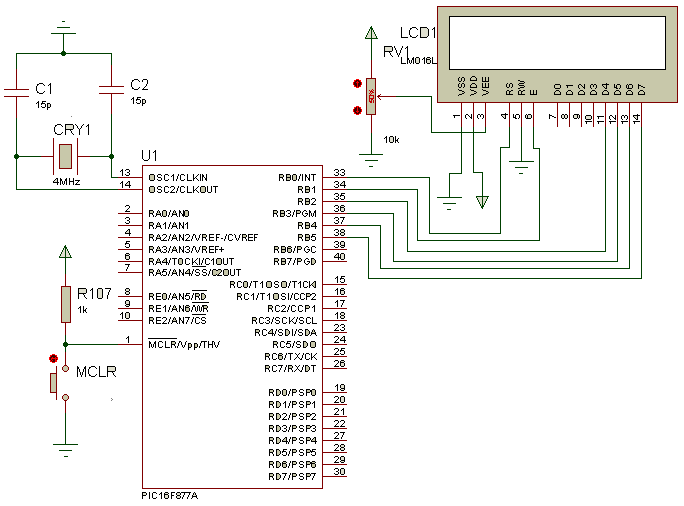


**0v**   **5V**

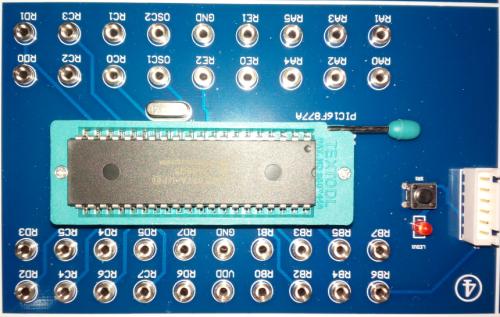
|  |  |
| --- | --- |
| **Programme N°1** | **Programme N°2** |
| **program compteur10;**  **begin**  **Trisb:=0;**  **portb:=0;**  **while (1=1) do**  **begin //segments [ g f e d c b a]**  **portb:=192 ; // chiffre 0**  **delay\_ms(1000); // attente d'une seconde**  **portb:=249; // chiffre 1**  **delay\_ms(1000);**  **portb:=164; // chiffre 2**  **delay\_ms(1000);**  **portb:=176; // chiffre 3**  **delay\_ms(1000);**  **portb:=153; // chiffre 4**  **delay\_ms(1000);**  **portb:=146; // chiffre 5**  **delay\_ms(1000);**  **portb:=130; // chiffre 6**  **delay\_ms(1000);**  **portb:=248; // chiffre 7**  **delay\_ms(1000);**  **portb:=128; // chiffre 8**  **delay\_ms(1000);**  **portb:=144; // chiffre 9**  **delay\_ms(1000);**  **end;**  **end.** | **program COMPTEUR2;**  **var i : integer;**  **const chiffre : array[10] of byte =(192, 249, 164,176, 153, 146, 130, 248, 128,144);**  **begin**  **Trisb:=0;**  **portb:=0;**  **while (1=1) do**  **begin for i := 0 to 9 do // compteur**  **begin**  **portb:=chiffre[i]; // affichage du chiffre**  **delay\_ms(1000); // attente d'une seconde**  **end;**  **end;**  **end.** |

## **Activité n° 3: Gestion d’un afficheur LCD**

Soit le montage suivant :



|  |
| --- |
| **Programme N°1** |
| **Program affichage ;**  **// Connections du module Lcd**  **var LCD\_RS : sbit at PORTB.0;**  **var LCD\_EN : sbit at PORTB.1;**  **var LCD\_D4 : sbit at PORTB.2;**  **var LCD\_D5 : sbit at PORTB.3;**  **var LCD\_D6 : sbit at PORTB.4;**  **var LCD\_D7 : sbit at PORTB.5;**  **var LCD\_RS\_Direction : sbit at TRISB.0;**  **var LCD\_EN\_Direction : sbit at TRISB.1;**  **var LCD\_D4\_Direction : sbit at TRISB.2;**  **var LCD\_D5\_Direction : sbit at TRISB.3;**  **var LCD\_D6\_Direction : sbit at TRISB.4;**  **begin**  **LCD\_init();**  **LCD\_CMD(\_LCD\_CURSOR\_OFF);**  **while true do**  **begin**  **LCD\_out(1,1,'FORMATION PIC');**  **end;**  **end.** |

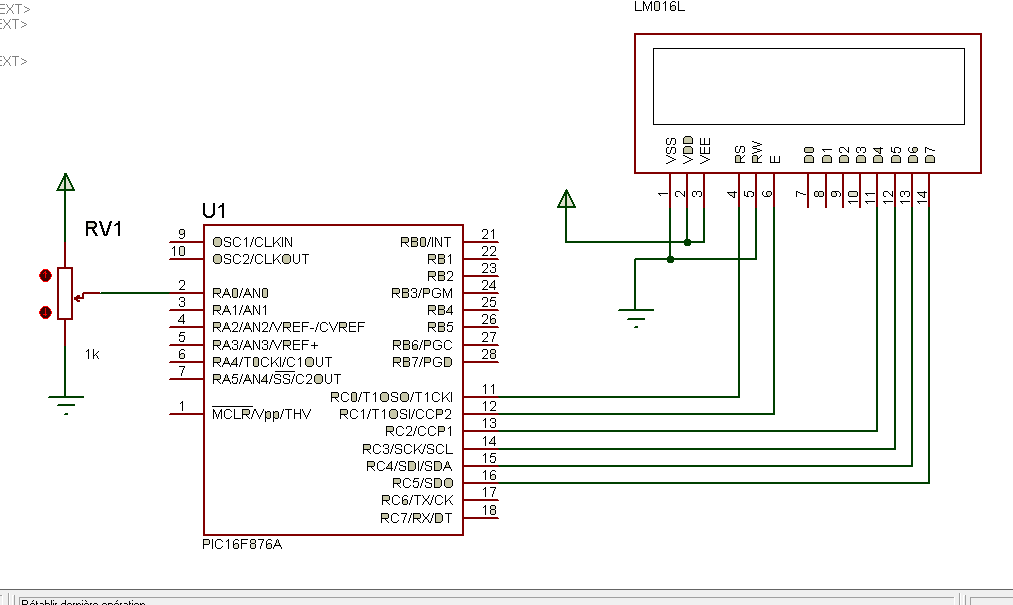


**0V 5V RB0 RB1 RB2……….RB5**

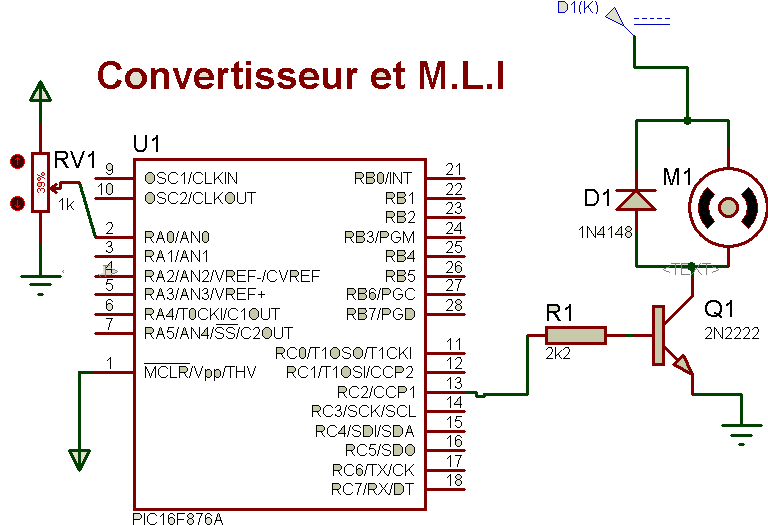
## **Activité n° 4: CAN - Voltmètre**

On désire de réaliser un voltmètre en utilisant CAN et afficheur LCD

|  |
| --- |
| **Programme** |
| **program voltmetre;**  **var**  **N : word ;**  **Cal : real ;**  **T: word;**  **valeur\_affichage : string[3];**  **LCD\_RS : sbit at portc.0;**  **LCD\_EN : sbit at portc.1;**  **LCD\_D4 : sbit at portc.2;**  **LCD\_D5 : sbit at portc.3;**  **LCD\_D6 : sbit at portc.4;**  **LCD\_D7 : sbit at portc.5;**  **LCD\_RS\_Direction : sbit at TRISC.0;**  **LCD\_EN\_Direction : sbit at TRISC.1;**  **LCD\_D4\_Direction : sbit at TRISC.2;**  **LCD\_D5\_Direction : sbit at TRISC.3;**  **LCD\_D6\_Direction : sbit at TRISC.4;**  **LCD\_D7\_Direction : sbit at TRISC.5;**  **begin**  **adcon1:=%10000000;**  **lcd\_init();**  **lcd\_cmd(\_LCD\_CURSOR\_OFF);**  **lcd\_out(1,1,'V=');**  **adc\_init();**  **while true do**  **begin**  **N :=adc\_read(0);**  **Cal := (N\*5000)/1023;**  **T:= word(Cal );**  **wordToStr(T,valeur\_affichage);**  **lcd\_out(1,3,valeur\_affichage);**  **lcd\_out(1,10,'mV');**  **delay\_ms(100);**  **end; end.** |



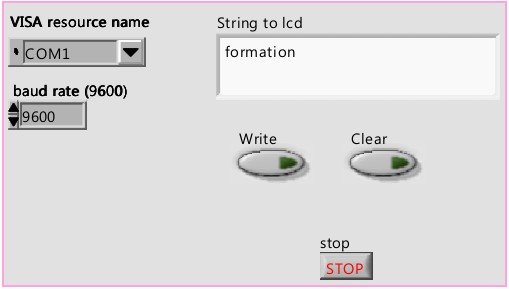
## **Activité n° 5: MLI – CONVERSIO N**

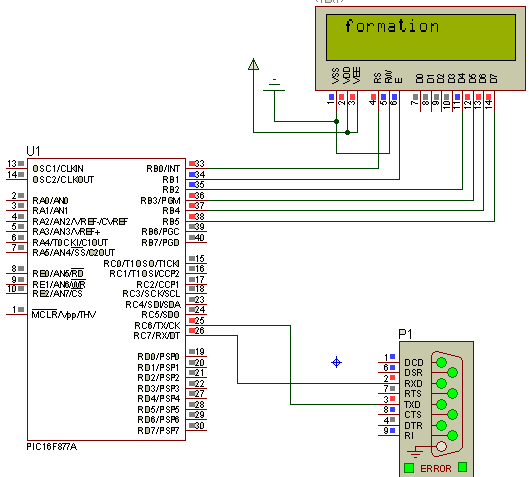


|  |
| --- |
| **Programme** |
| **program MLI\_conversion;**  **var**  **K : byte;**  **N : word ;**  **begin**  **PWM1\_init(250); // Initialiser le module PWM1 et choix de la fréquence de PWM1= 250 Hz**  **ADCON1:=$80; // $8E Configuration des entrées du porta comme entrées analogiques y compris RA0**  **PWM1\_start; // démarrage du module PWM1**  **while true do // boucle infinite**  **begin**  **N:= adc\_read(0); // lecture de la conversion**  **K:=N/4 ; // calcul**  **PWM1\_set\_duty ( K ); // changement du rapport cyclique : le rapport cyclique est K/ 255**  **end;**  **end.** |

## **Activité n° 6: Interface UART + LABVIEW**

On désire de créer une interface en **LABVIEW** pour commander un afficher LCD :





|  |
| --- |
| **Programme** |
| **program testlcd;**  **var i : char;**  **var LCD\_RS : sbit at RB0\_bit;**  **var LCD\_EN : sbit at RB1\_bit;**  **var LCD\_D4 : sbit at RB2\_bit;**  **var LCD\_D5 : sbit at RB3\_bit;**  **var LCD\_D6 : sbit at RB4\_bit;**  **var LCD\_D7 : sbit at RB5\_bit;**  **var LCD\_RS\_Direction : sbit at TRISB0\_bit;**  **var LCD\_EN\_Direction : sbit at TRISB1\_bit;**  **var LCD\_D4\_Direction : sbit at TRISB2\_bit;**  **var LCD\_D5\_Direction : sbit at TRISB3\_bit;**  **var LCD\_D6\_Direction : sbit at TRISB4\_bit;**  **var LCD\_D7\_Direction : sbit at TRISB5\_bit;**  **begin**  **Lcd\_Init(); // Initialize LCD**  **Lcd\_Cmd(\_LCD\_CLEAR); // Clear display**  **Lcd\_Cmd(\_LCD\_CURSOR\_OFF); // Cursor off**  **uart1\_init (9600); // init uart**  **delay\_ms (100);**  **while true do**  **begin**  **if ( uart1\_data\_ready() <> 0 ) then**  **begin**  **i:= uart1\_read();**  **if ( i = 13 ) then**  **lcd\_cmd (\_lcd\_clear);**  **LCD\_chr\_cp(i);**  **end;**  **end;**  **end.** |

