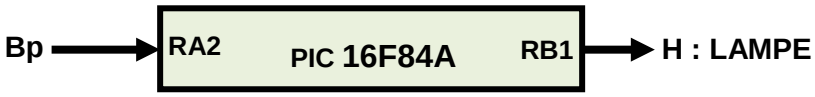
GENIE ELECTRIQUE Prof : Mr Raouafi Abdallah	Série d'exercices N°10 « Microcontrôleurs »	Niveau : 4ème Sc.Technique

Exercice n°1: Circuit simple allumage

Cahier des charges : un bouton poussoir (**Bp**) connecté à la broche **RA2** du **PORT.A** commande une **LAMPE (H)** connectée à la broche **RB1** du **PORT.B**.



1. Déterminer les mots binaires et hexadécimaux à installer dans les registres **Tris A** et **Tris B.**

Tris A	--	--	--	RA4	RA3	RA2	RA1	RA0
				.....	.....	.....	.....	.....
Tris B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	.....	.....	.....	.....	.....	.....	.....	.....

Tris A = (.....)$_2$ = (.....)$_{16}$

Tris B = (.....)$_2$ = (.....)$_{16}$

2. Ecrire l'algorithme et traduire en programme Mikropascal.

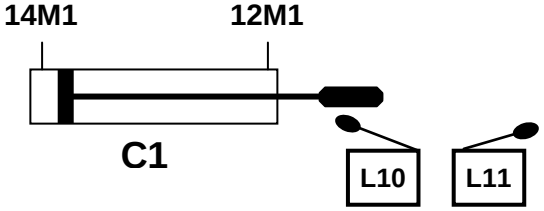
Algorithme	Programme

Algorithme ALLUMAGE_LAMPE ;
Variable :
Bp : bit du registre PORT A affecté à RA2 ;
H : bit du registre PORT B affecté à RB1 ;
Début
Tris A ⇐ % ..... ;
Tris B ⇐ % ..... ;
Port ..... ⇐ $ ..... ;
TANQUE (1=1) FAIRE // boucle infinie
Début
Si (Bp=1) Alors H⇐..... Si non H⇐..... ;
Fin ;
Fin. // fin programme
	Program ALLUMAGE_LAMPE ;
	Var :
	Bp : sbit at RA2_bit ;
	H : ..... ;
	Begin
	Tris A := $ ..... ;
	Tris B := $ ..... ;
	PORT B := $ ..... ;
	while (true) do // boucle infinie
	begin
	if (Bp = 1) then H:= ..... else H:= ..... ;
	.....
	end. // fin programme

Exercice n°2 : Grafcet va et vient

Avec un seul vérin on veut réaliser un **cycle va et vient** à base du **PIC 16F84A.**

Entrées système	Entrées PIC	Sorties système	Sorties PIC
Dcy	RA0	12M1	RB1
L10	RA2	14M1	RB3
L11	RA4		



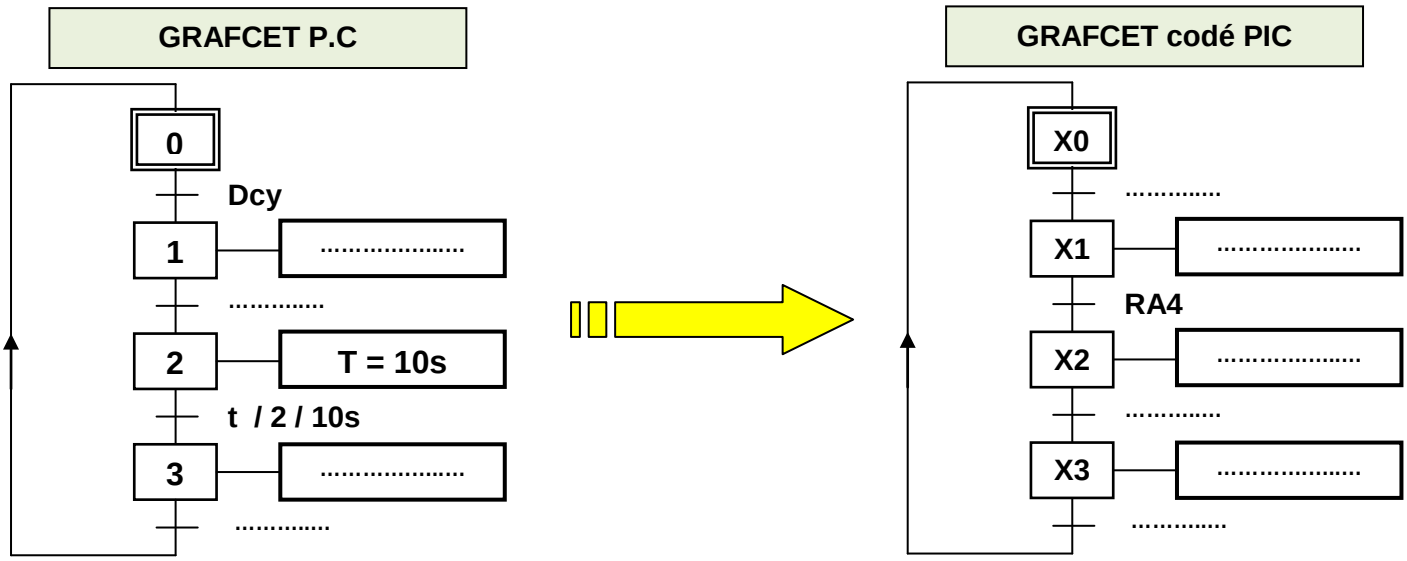
1. Déterminer les mots binaires et hexadécimaux à installer dans les registres **Tris A** et **Tris B.**

Tris A	--	--	--	RA4	RA3	RA2	RA1	RA0
				.....	.....	.....	.....	.....
Tris B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	.....	.....	.....	.....	.....	.....	.....	.....

Tris A = (.....)$_2$ = (.....)$_{16}$

Tris B = (.....)$_2$ = (.....)$_{16}$

2. Déterminer le Grafcet PC et traduire en Grafcet codé PIC 16F84A.



3. Ecrire l'algorithme et traduire en programme Mikropascal.

Algorithme	Programme
Algorithme G7cycle_v ;	**Programm G7cycle_v;**
Variable :	**Var :**
Dcy : bit du registre PORT A affecté à RA0 ;	**Dcy : Sbit at RA0 _ bit ;**
L10 : bit du registre PORT A affecté à …… ;	**L10 : Sbit at ……. _ bit ;**
L11 : bit du registre PORT A affecté à …… ;	**L11 : Sbit at ……. _ bit ;**
12M1 : bit du registre …… affecté à ……. ;	**12M1 : Sbit at ……. _ bit ;**
14M1 : bit du registre …… affecté à ……. ;	**14M1 : Sbit at ……. _ bit ;**
X0, X1, X2, X3 : bit mémoire ; // étapes G7	**X0, X1, X2, X3 : bit ; // étapes G7**
T : bit ; // temporisateur T	**T : bit ; // temporisateur T**
DEBUT	**BEGIN**
Tris A ⇐ $ ..... ;	tris A := % ……..... ;
Tris B ⇐ % ..... ;	tris B := $ ..... ;
12M1⇐0 ; 14M1⇐0 ; T⇐0 ;	**12M1 := 0 ; 14M1 := 0 ; T := 0 ;**
// état initial des sorties du GRAFCET	**// état initial des sorties du GRAFCET**
X0⇐..... ; X1⇐..... ; X2⇐..... ; X3⇐..... ;	**X0 := 1 ; X1 := 0 ; X2 := 0 ; X3 := 0 ;**
// état initial des étapes du GRAFCET	**// état initial des étapes du GRAFCET**
TANQUE (1=1) FAIRE	**While (true) do**
Début	**begin**
////// Programmation des étapes ///////////	**////// Programmation des étapes ///////////**
Si (X0=1) ET (……) Alors	**if (X0=1) and (……) then**
Début	**begin**
X0⇐…… ; X1⇐…… ; Fin si ;	**X0:=…… ; X1:=…… ; ……. ;**
Si…… Alors	**if (……) and (……) then**

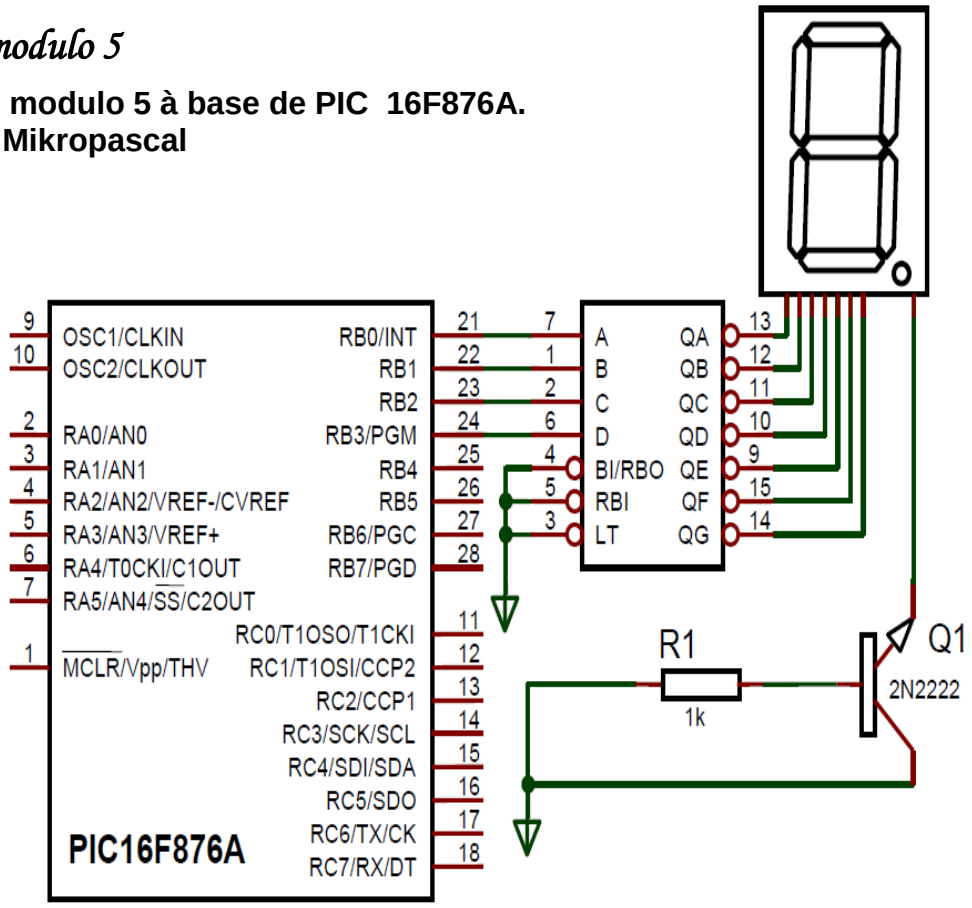
Début
…… ; …… ; Fin si ;
Si…… Alors
Début
…… ; …… ; Fin si ;
Si…… Alors
Début
…… ; …… ; Fin si ;
////////// Traitement des sorties //////////
Si **X1=1** alors 14M1 ⇐…. si non 14M1 ⇐…. ;
Si **X2=1** alors T ⇐…. si non T ⇐…. ;
Si **X3=1** alors 12M1 ⇐…. si non 12M1 ⇐…. ;
////////// Traitement de temporisateur //////////
Si **(T=1)** alors **Attente (10s) ;**
Fin ;
Fin.
	begin
	…… ; …… ; …… ;
	if (……) and (……) then
	begin
	…… ; …… ; end ;
	if (……) and (……) then
	begin
	…… ; …… ; end ;
	////////// Traitement des sorties //////////
	if **X1=1** then **…… := ….** else **…… := …. ;**
	if **X2=1** then **…… := ….** else **…… := …. ;**
	if **X3=1** then **…… := ….** else **…… := …. ;**
	////////// Traitement de temporisateur //////////
	if **(T=1)** then **delay_ms (……) ;**
	…… ;
	END.

Exercice n°3 : *Compteur modulo 5*

Soit le montage de comptage modulo 5 à base de PIC 16F876A.

1. **Compléter le programme Mikropascal**

```
Program COMPTEUR-5 ;
Var
N : …… ; // (N est sur 8 bits)
Begin
trisb:=$…… ;
      // RB0 à RB3 : sorties
while (true) do
   Begin
   N:=0;
   While (N < …..) Do
     Begin
      Portb:=N; // N reçoit la
                valeur du portB
      Delay_ms(……..) ; // 1s
      N := ……. ; // comptage
      End ;
    End ;
End .
```



Tris B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	…..	…..	…..	…..	…..	…..	…..	…..

2. Configurer avec timer (TMRO) en mode compteur modulo 5.

Rappel sur la registre : OPTION_REG

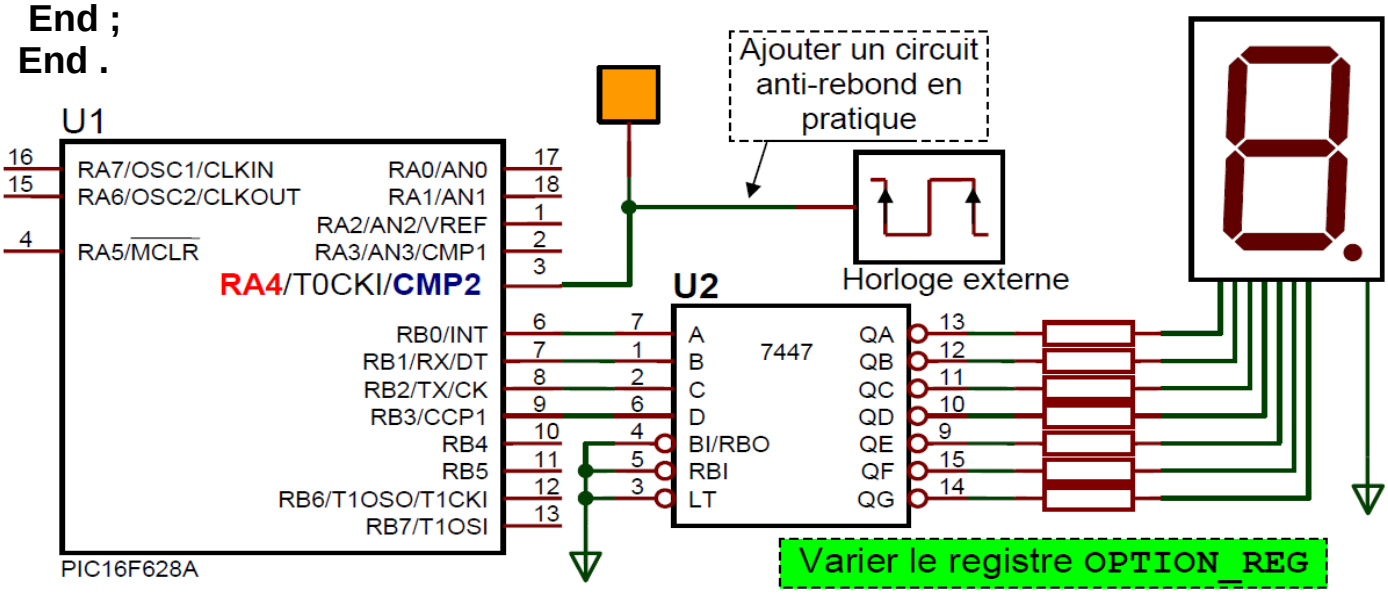
/ RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1 (act RA4)	0 (montant)	1 (front)	0	0	0

bit 7	/RBPU : bit de validation des résistances de pull-up du port B 1 = Résistances de pull-up du port B désactivées (Pas de résistances) 0 = Résistances de pull-up du port B (Résistances de tirage reliées à +VDD)
bit 6	INTEDG : bit de sélection de front d'interruption 1 = interruption sur front montant de la pin RB0/INT 0 = interruption sur front descendant de la pin RB0/INT
bit 5	T0CS : bit de sélection de la source d'horloge du timer0 1 = sur transition de la pin RA4/T0CKL 0 = sur horloge interne cycle d'instruction (CLKOUT)
bit 4	T0SE : bit de sélection de source de front pour Timer0 1 = incrémentation sur front descendant de la pin RA4/T0CKL 0 = incrémentation sur front montant de la pin RA4/T0CKL
bit 3	PSA : Prescaler assignement bit = bit d'assignation du pré-diviseur 1 = Pré-diviseur assigné au WDT (Watch Dog Timer) [A chaque front on ajoute 1 COMP] 0 = Pré-diviseur assigné au module Timer0 (Tableau pré diviseur ci-dessous)
Bits 2 - 1 - 0	PS2 :PS0 : bits de sélection du ratio de pré-diviseur (voir tableau Pré diviseur ci-contre :

PS2	PS1	PS0	Pré-diviseur
0	0	0	1 : 2
0	0	1	1 : 4
0	1	0	1 : 8
0	1	1	1 : 16
1	0	0	1 : 32
1	0	1	1 : 64
1	1	0	1 : 128
1	1	1	1 : 256

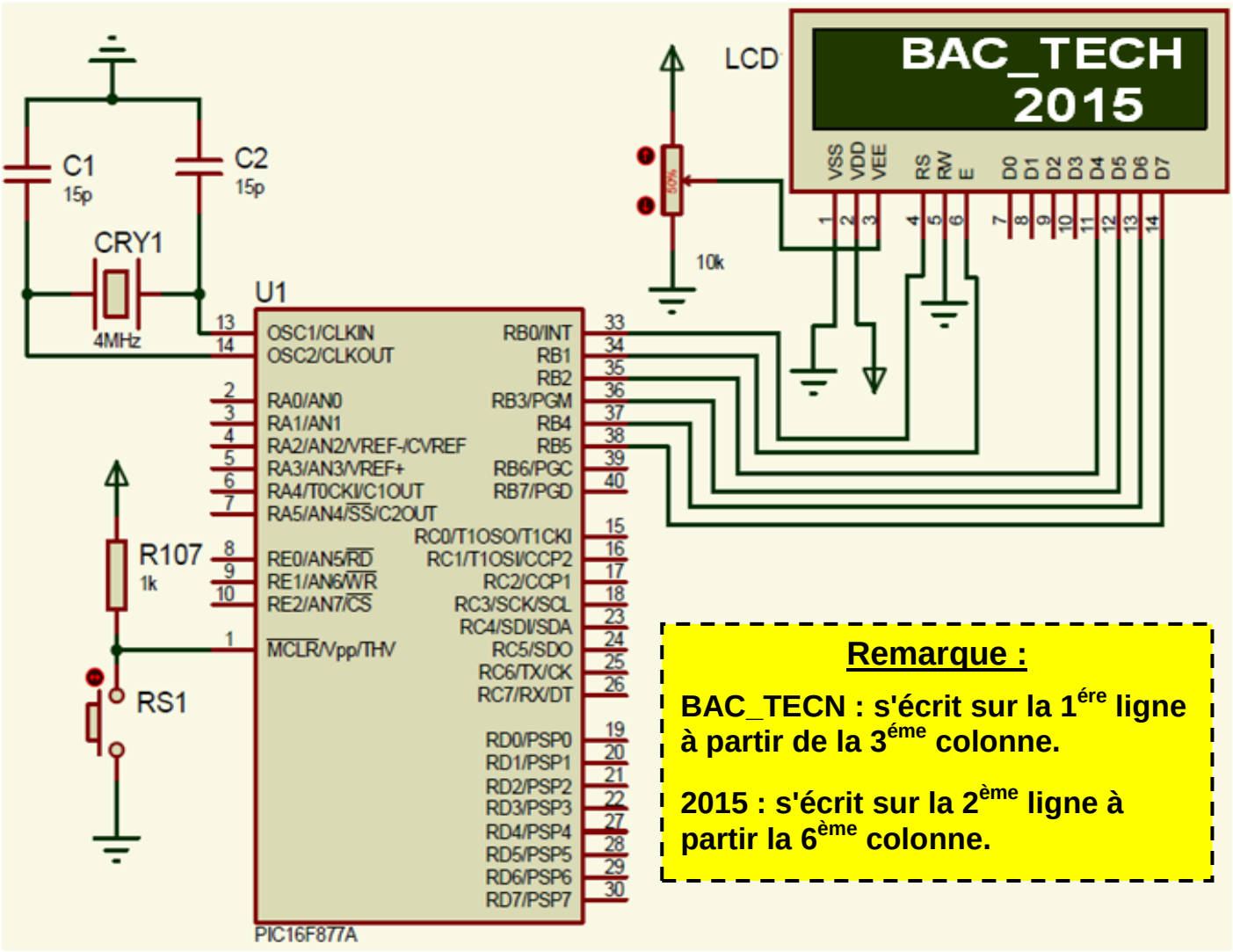
Compteur modulo 5 avec RA4 :

```
program compteur 5_RA4; // compteur modulo 5
begin
trisa:=$..... ; // le port A est configuré en entrées
trisb:=$..... ; // le port B st configuré en entrées/ sorties
CMCON := $ 07; // Désactiver les comparateurs
OPTION_REG := % ..... ; // à choisir selon le
                          cahier des charges
While (true) do // boucle répétitive sans limite
Begin
 TMR0:= ..... ; // initialiser le timer
 While TMR0 < ..... do      // Compteur modulo 5
 Portb:=TMR0 ; // le contenu du timer affiché sur le port B
 End ;
End .
```



Exercice n°4 : *Commande d'un afficheur LCD*

Compléter le programme permettant d'écrire « BAC_TECH 2015 » en utilisant PIC 16F877A :



Remarque :

BAC_TECN : s'écrit sur la 1ére ligne à partir de la 3éme colonne.

2015 : s'écrit sur la 2ème ligne à partir la 6ème colonne.

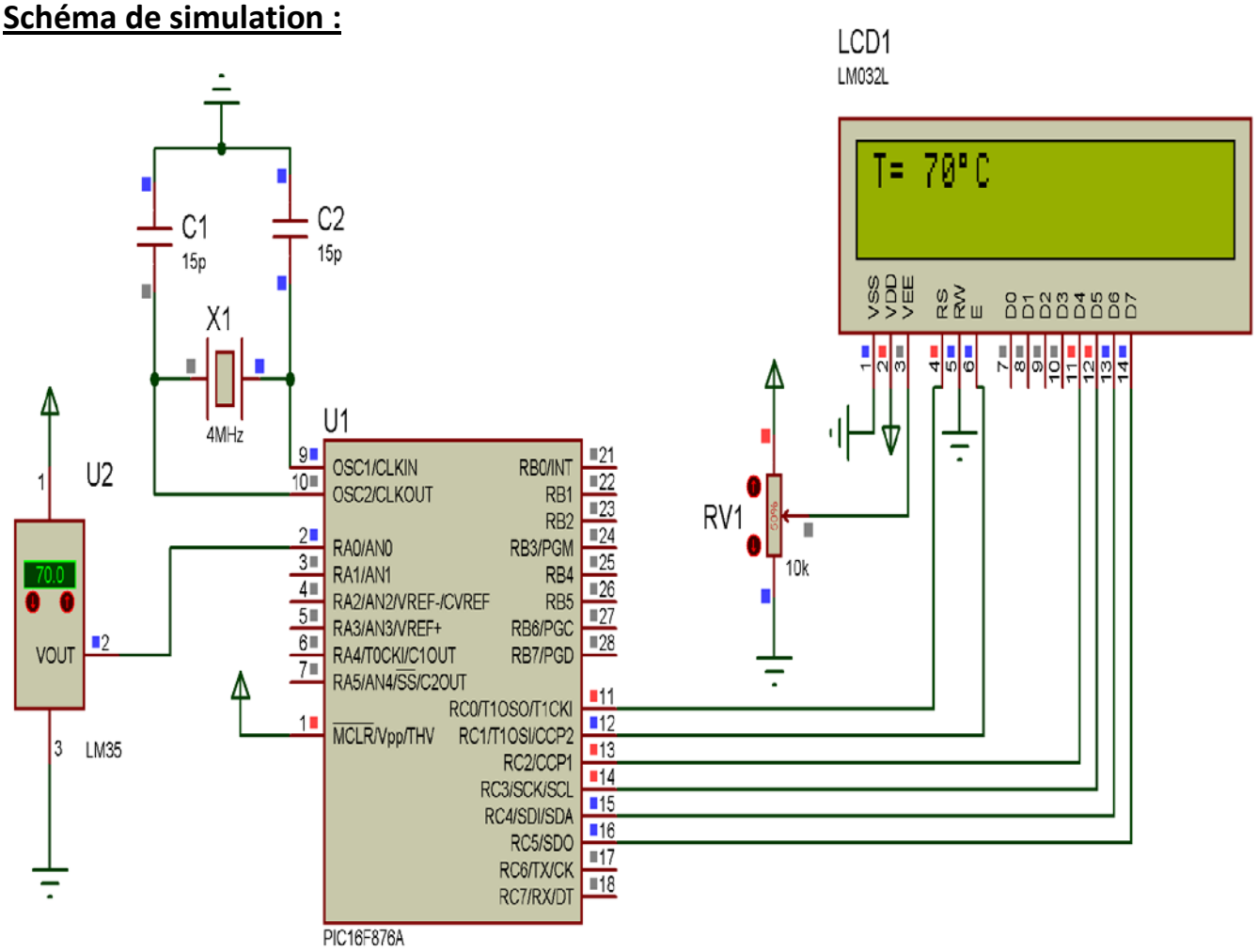
```
Program  affichage ;
// connections du module LCD
Var  LCD_RS : sbit at PORTB.0 ;
Var  LCD_EN : sbit at …… ;
Var  LCD_D4 : sbit at …… ;
Var  LCD_D5 : …… ;
Var  LCD_D6 : …… ;
Var  LCD_D7 : …… ;
Var LCD_RS_Direction : sbit at TRISB.0 ;
Var LCD_EN_Direction : sbit at …… ;
Var LCD_D4_Direction : sbit at …… ;
Var LCD_D5_Direction : sbit at TRISB.3 ;
Var LCD_D6_Direction : sbit at ……
Var LCD_D7_Direction : ……
```

```
Begin
LCD_init ( ) ; // Routine qui initialise le LCD
LCD_CMD (_LCD_CURSOR_OFF) ;
      // supprimer le curseur de LCD
 While (true) do
    Begin
    LCD_out ( …… , …… ,' BAC_TECN ') ;
    LCD_out ( …… , 6 ,'……') ;
    End ;
End .
```

Exercice n°5: *Convertisseur analogique numérique (C.A.N)*

La plupart des sondes de températures dans l'automobile sont de types à résistance variable. Le paramètre physique qui varie en fonction de la température est la résistance du matériau qui constitue la sonde. Dans notre cas on utilise un microcontrôleur PIC16F876A, un afficheur LCD et un capteur de température LM35.

- **Schéma de simulation :**



Le capteur LM35 fournit à sa sortie une tension proportionnelle à la température (10mV/°C), en appliquant la règle de 3 on obtient : T = ……

Le convertisseur **CAN** du microcontrôleur **PIC16F876A** converti toute tension comprise entre **0** et **5V** (**0** à **5000 mV**) en un nombre sur **10 bits** (de **0** à 2^{10}**-1** donc de **0** à **1023**).

En appliquant la règle de 3 on obtient :

$$V = \frac{N \times 5000}{1023} \quad \{ V \text{ en } mV \quad \Rightarrow \quad T = \frac{N \times 500}{1023} \quad \{ T \text{ en } °C$$

****

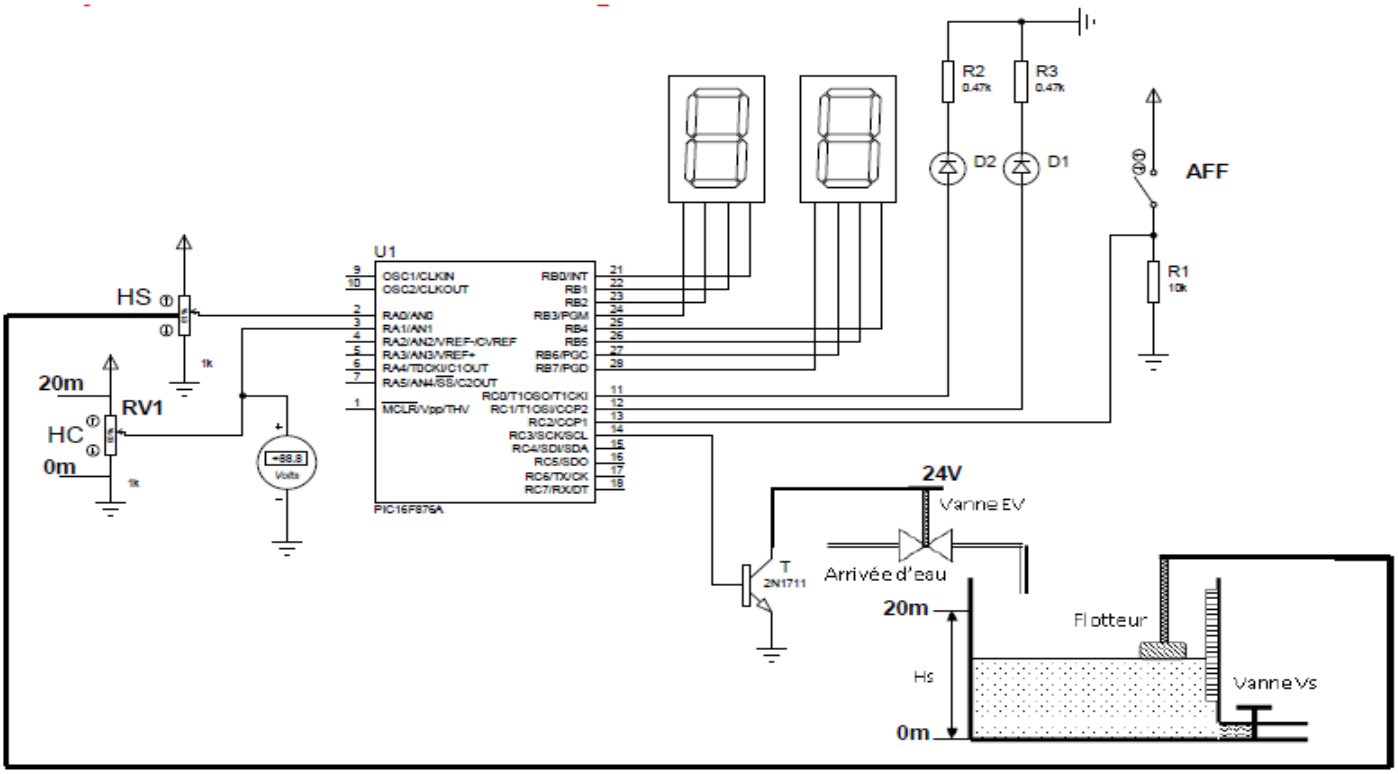
Ecrire un programme qui permet de lire et afficher la température sur l'afficheur LCD :

****

```
Program  conversion CAN ;
Var
Valeur_conversion : word ;   // N sur 2 octets car le résultat de conversion est sur 10 bits
Variable_calcul : real ;  // V : type réel  pour le calcul afin ne pas avoir un dépassement de
                              taille lors de la multiplication ou une perte de précision lors
                              de la division
Temperature : byte ;     // T sur 1 octet car la température est comprise entre 2 et 150
Valeur_afichage : string [3] ;    //chaine de 3 caractères pour afficher la température
        // Connections de l'LCD
        LCD_RS : sbit  at  port c.0 ;
        LCD_EN : sbit  at  port c.1 ;
        LCD_D4 : sbit  at ……..
        LCD_D5 : sbit  at ……..
        LCD_D6 : sbit  at ……..
        LCD_D7 : sbit  at ……..
        LCD_RS_Direction : sbit  at  TRIS c.0 ;
        LCD_EN_Direction : sbit ……..
        LCD_D4_Direction : sbit ……..
        LCD_D5_Direction : sbit ……..
        LCD_D6_Direction : sbit  at  TRIS c.4 ;
        LCD_D7_Direction : sbit ……..
Begin
   Lcd_init () ; // initialisation de l'LCD
   Lcd_cmd (_LCD_CURSOR_OFF); // désactivation du curseur de l'LCD
   Lcd_out (1,1,'T=');       // préparation de l'affichage
   adcon1:=%10000100 ;       // choix de RA0/AN0 en tant que entrée analogique
   adc_init ();              // initialisation du module CAN
While true do
  Begin
    valeur_conversion := adc_read (0);  // Lecture du convertisseur
    variable_calcul  := (valeur_conversion * …….. ) / ……. ;   // Calcul
    temperature:= byte (variable_calcul ); // transformation en octet
    byteTo Str (temperature,valeur_afichage);      // conversion de la température en texte
    Lcd_Out (1,3,Valeur_Afichage);     // Affichage de la valeur de la température
    Lcd_Chr (1,6,%11011111);        // Affichage du symbole degré : °
    Lcd_Chr (1,7,'C');              // affichage de C pour Celsius
    Delay_ms (600);                 // attente de 600ms puis rafraichissement de l'affichage
  End;
End.
```

Exercice n°6: *Château d'eau commandé par le PIC 16F876A.*

On donne le montage de principe suivant et on demande de compléter le programme Mikropascal :



- **RV1** : potentiomètre permettant de fixer la hauteur de consigne.
- **Aff =0** : autorise l'affichage de la hauteur de niveau d'eau
- **Aff =1** : autorise l'affichage de la valeur de la hauteur consigne.
- **D1**: signal que la valeur affichée est celle de la hauteur consigne.
- **D2** : signal que la valeur affichée est celle de la hauteur de niveau d'eau.

```
programC_temp;
var
 Aff: sbit at RC2_bit; //Commutateur
 LED1:sbit at RC1_bit;
 LED2:sbit at RC0_bit;
 Commande_EV: sbit at RC3_bit;
Nc , Ns : real;
Hc,Hcd,Hcu,Hs,Hsu,Hsd : byte ;
begin
ADCON1 :=$80 ; //Configuration du registre
      trisa :=$ff ;
❶Trisb: =$.....;
❷Trisc: =$.....;
while true do
begin
//Consigne hauteur
❸Nc := adc_read (.....);
// hauteur de niveau d'eau
❹Ns := adc_read(....);
Hs := (Ns * ......)/1023 ;
Hsu:=(Hs mod 10);  // Unités
Hsd:=(Hs div 10);  //Dizaines
```

```
//Consigne hauteur
❺Hc:= ..................................................;
Hcu:=(Hc mod 10);  // Unités
Hcd:=(Hc div 10);  // Dizaines
// Affichage
if Aff=1 then
begin
portb:= Hcd + Hcu *.....;// décalage à gauche
 de 4 de l'unité
LED1:=1;
LED2:=0;
end
else
begin
portb:= ..... + Hsu *.....;
LED1:=0;
LED2:=1;
end;
// Commande de l'électrovanne d'entrée
❻if .....then Commande_EV:=1;
❼if .....then Commande_EV:=0;
end; End.
```