AUTOMATIQUE

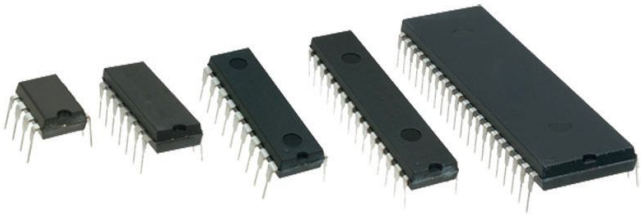
Leçon : A3-2

Les microcontroleurs

Objectifs :
- *Décrire le fonctionnement d'un système par un algorithme.*
- *Traduire un algorithme en programme écrit en langage évolué.*
- *Elaborer un programme spécifique à une application à base de PIC.*
- *Transférer un programme vers un microcontrôleur.*

I- Rappels sur les microcontrôleurs:

1- Définition :



Un microcontrôleur est un circuit intégré programmable.

Comme un ordinateur, il est composé essentiellement d'un Microprocesseur, de mémoires et de périphériques.

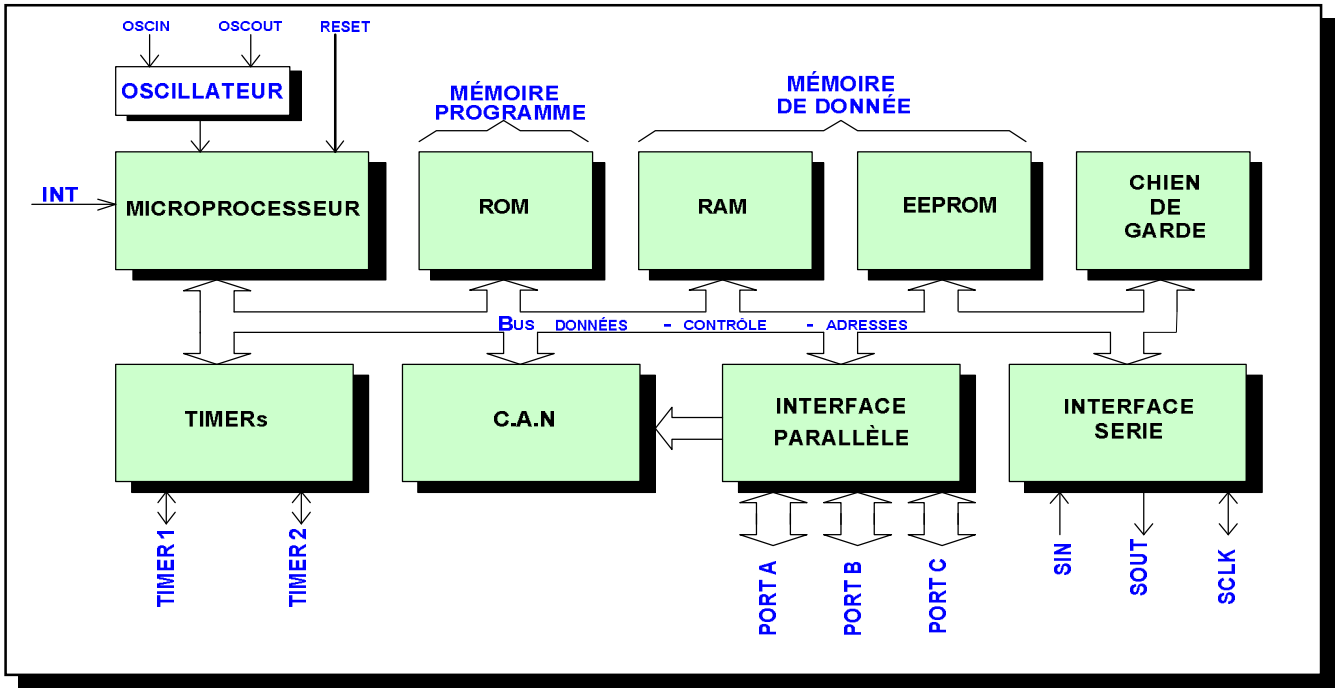
Il effectue principalement des opérations d'arithmétiques et des opérations logiques.

Donc il fonctionne de façon autonome au sein d'une application.

2- Structure interne :

Généralement on trouve à l'intérieur d'un microcontrôleur :

- Un **microprocesseur** (C.P.U.),
- De **la mémoire de donnée** (RAM et EEPROM),
- De **la mémoire programme** (ROM, OTPROM, UVPROM ou EEPROM),
- Des **interfaces parallèles** pour la connexion des entrées / sorties,
- Des **interfaces séries** (synchrone ou asynchrone) pour le dialogue avec d'autres unités,
- Des **timers** pour générer ou mesurer des signaux avec une grande précision temporelle,
- Des **convertisseurs analogique / numérique** pour le traitement de signaux analogiques.



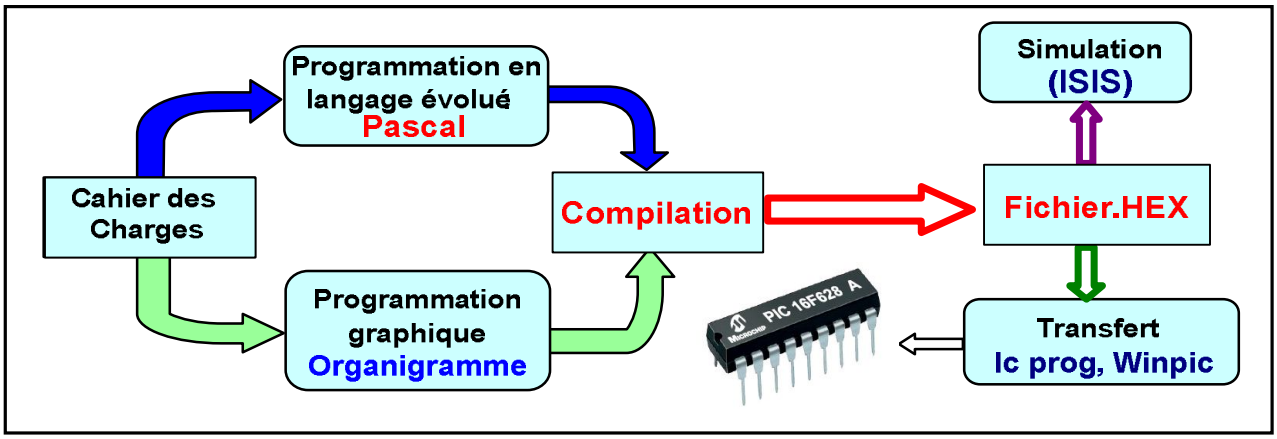
3- Caractéristiques :

Caractéristiques	16F628A	16F876A	16F877A
Mémoire programme (octets)	2048	8192	8192
Entrées/Sorties	16 (2ports)	22 (3ports)	33 (5ports)
Fréquence d'horloge (MHz)	20	20	20
Sources d'interruptions	10	14	15
Timer (TMR)/Compteur	TMR0 (8 bits) TMR1(16bits) TMR2 (8 bits)	TMR0 (8 bits) TMR1 (16 bits) TMR2 (8 bits)	TMR0 (8 bits) TMR1 (16 bits) TMR2 (8 bits)
Convertisseur analogique/numérique (ADC)	-	1 ADC 10bits 5 canaux	1 ADC 10bits 8 canaux
CCP	1	2	2
Comparateur analogique	2	2	2
Nombre des broches	18	28	40

4- Les registres de configuration et de contrôle :

TRISx								
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0

5- Les étapes à suivre pour programmer un microcontrôleur :



6- Programmation d'un PORTX :

Le registre **TRISX** est utilisé pour configurer le **PORTX** en **entrée ou en sortie**

Un bit à **0** configure la broche correspondante en **sortie (Out).**

Un bit à **1** configure la broche correspondante en **entrée (In).**

Exemple : On écrit dans le registre **TRISB** la valeur binaire **10010111**, le tableau suivant donne la configuration du port B correspondante.

TRISB								
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

TRISB =……$_{(2)}$ **=**……$_{(16)}$ **=**……$_{(10)}$

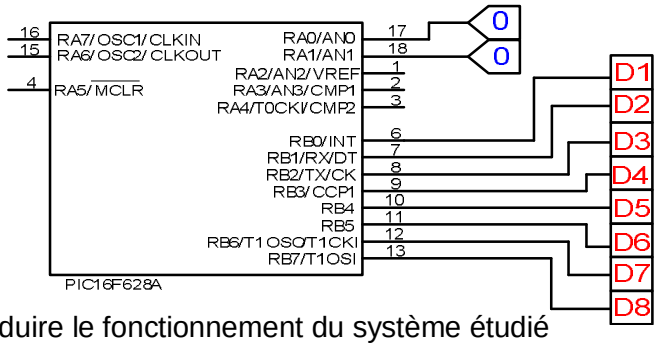
II- Applications à base de PIC :

Activité N°1 :

La signalisation lumineuse du robot aspirateur est assurée par une série de 8 diodes LED de couleur rouge.
Ces diodes permettent à l'utilisateur de connaitre l'état du robot à distance sans être obligé de s'approcher pour lire l'affichage LCD.
Le fonctionnement des diodes est le suivant :

- A l'arrêt les diodes sont toutes éteintes
- En fonctionnement le robot allume les diodes l'une après l'autre (chenillard)
- Lorsque sa batterie est à la limite de la décharge, le robot effectue un clignotement des diodes.
- Durant la charge de la batterie toutes les diodes sont constamment allumées.

PORTA	RA1	RA0	Etat des diodes
....	0	0	Eteintes
1	0	1	Clignotent
....	1	0	chenillard
....	1	1	Allumées



Compléter les programmes ci-dessous permettant de traduire le fonctionnement du système étudié

1ére méthode :

```
Program ACT1M1;
VAR  ……..
begin
  trisa:= ……..
  trisb:= ……..
  portb:= ……..
  while …….  do
   begin
    If (porta=0)    then ……..
    If  porta=3     then ……..
    If  porta=1     then
                        begin
                  ……..
                  ……..
                        end;
    If  porta=2    then
       begin
             portb:= ……..
          for i:= …… to …… do
                 begin
                    delay_ms(500);
                    portb:= portb SHR 1;
                 end;
        end;
   end;
end.
```

2éme méthode :

```
Program ACT1M2;
VAR        diodes: byte at portb;
           entrées: byte at porta;
Procedure marche (const a: byte);
 Begin
  ……..
 end;
Procedure clignotant();
        begin   ……..
                ……..
        end;
Procedure  chenillard();
var i:byte;
         begin    ……..
              for i:= ……. to ….. do
                    begin
                        delay_ms(500);
                        diodes:= diodes SHL 1;
                    end;
          end;
Begin
  trisb:=……..; trisA:= ……..
  While …….. do
   Begin
  IF entrées= 0 then  ……..
  IF entrées= 1 then  ……..
  IF entrées= 2 then  ……..
  IF entrées= 3 then  ……..
end;
end.
```

Activité N°2 : Programmation d'un GRAFCET

On se référant aux ***GRAFCET*** de point de vue PC et à la table d'affectation Compléter :

1- Le ***GRAFCET*** codé microcontrôleur.

2- Le schéma de simulation ci- dessous.

3- Le programme en langage ***Mikropascal***.

Tableau d'affectation

Entrées /Sorties Système	Broche du microcontrôleur
m	RB0
a	RB1
b	RB2
d	RB3
KM1	RA1
AV	RA2
AR	RA3

GRAFCET PC

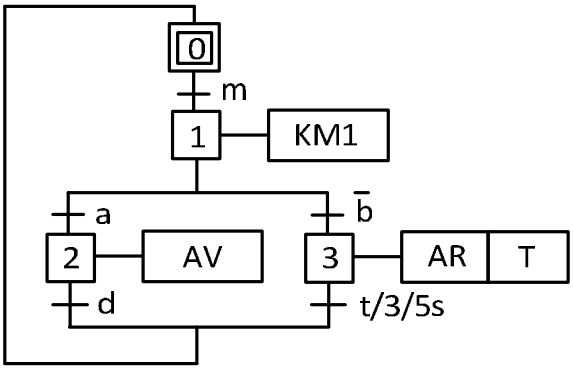
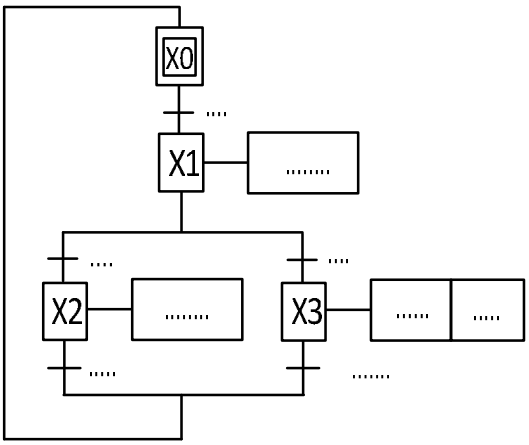
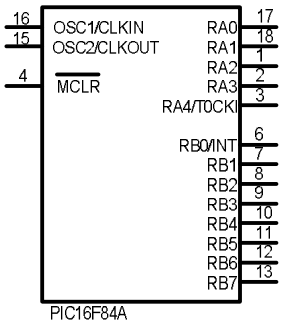
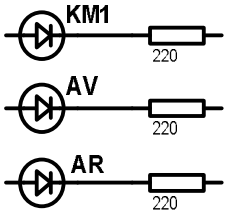


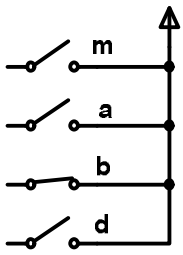
Schéma de simulation

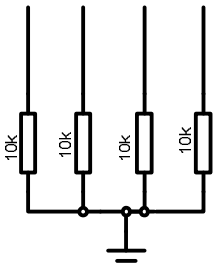
GRAFCET codé microcontrôleur











Programme en Mikropascal:

Program GRAFCET;

Var

m: sbit at RB0_bit;

a: ……

b: ……

…: sbit at RB3_bit;

KM1: ……

……..

……..

X0, ... : byte ;

begin

trisa:=.....

trisb:= .....

…… **// état initial des sorties**

…… **// état initial des étapes**

```
While ……. ……..          // boucle infinie
  begin
       if (……..……) and (……..……)   then  begin  ……..……..  end;
       if (……..……) and (……..……)   then   begin ……..……..
       if (……..……) and (……..……) then    ……..……..
       if ((……..……) and (……..……))  ….. ((……..……) and (……..……)) then
         ……..……..
       If  X1=1   then ……..……..
       If  X2=1   then ……..……..
       If  X3=1   then ……..……..
       if  t=1   then  delay_ms(……..……);
  end;
end.
```

Activité N° 3: Ecran LCD

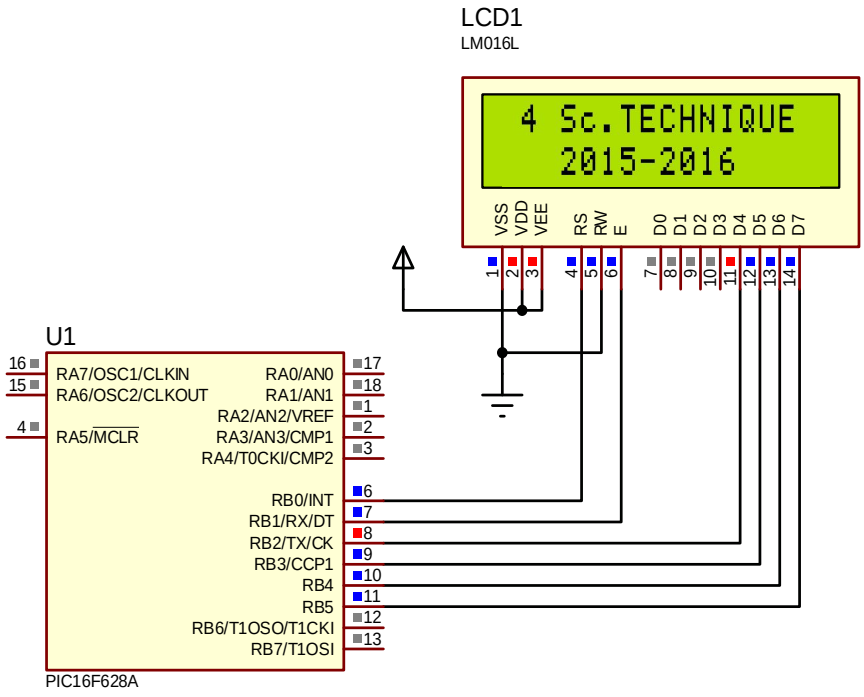
Sur un afficheur LCD « LM016L » écrire : • **4 Sc. TECNIQUE 1** sur la 1ère ligne et la 1ère colonne

• **2014- 2015** sur la 2ème ligne et la 4ème colonne

Compléter puis saisir le programme ci-contre puis simuler et vérifier le fonctionnement :

```
Program ACT3;
// Connections du module Lcd
Var
 LCD_RS : sbit at PORTB.0;
 LCD_EN : ……..;
 LCD_D4 : ……..;
 LCD_D5 : ……..;
 LCD_D6 : ……..;
 LCD_D7 : ……..;

 LCD_RS_Direction : sbit at TRISB.0;
 LCD_EN_Direction : ……..;
 LCD_D4_Direction : ……..;
 LCD_D5_Direction : ……..;
 LCD_D6_Direction : ……..;
 LCD_D7_Direction : ……..;
begin
  ……..
  LCD_CMD(_LCD_CURSOR_OFF);        // initialisation de LCD
  while true do                    // ……..
   begin
   ……..                            // 4 Sc. TECNIQUE  s'écrit à la 1ère ligne et la 2ère colonne
   ……..                            // 2015-2016 s'écrit à la 2ème ligne et la 4ème colonne
   end;
end.
```

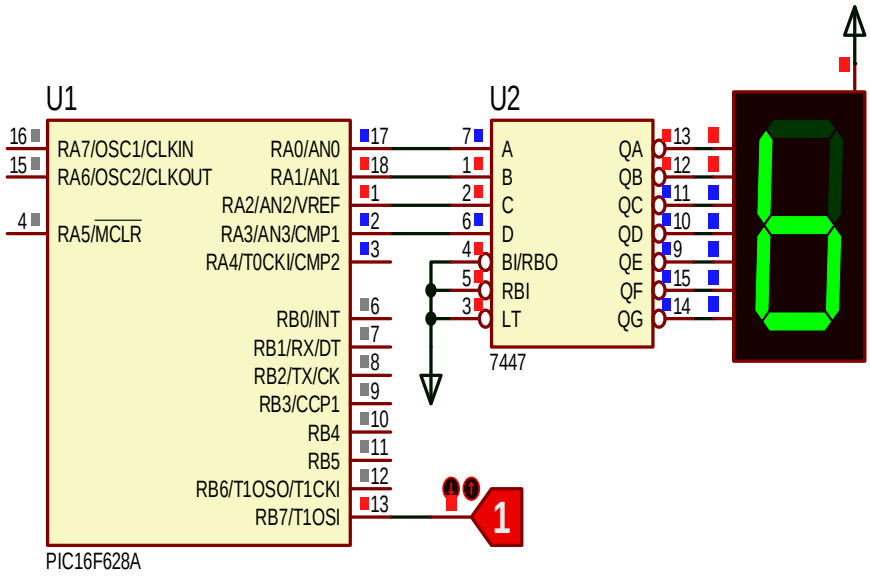


Activité N°4 : Compteur / décompteur modulo 10

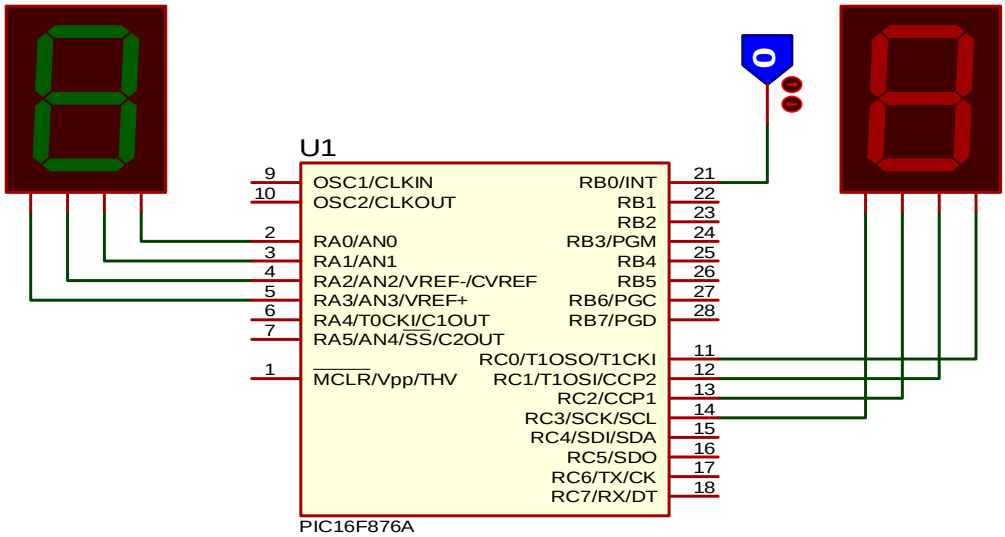
Compléter le programme suivant permettant d'avoir:

- ***un compteur modulo 10 si a est actionné (a=1)***
- ***un décompteur modulo 10 si a = 0***

```
Program ACT4;
Var   i:integer;
Begin
    trisb:=…….
    trisa:=…….
    porta:=…….
    CMCON:=$07;
while true do
  begin
     if portb.7=1 then
       begin
        for i:=0 ……..
          begin
          …….; delay_ms (500);
          end;
       end
       else
          begin
           for i:= ……..
            ……
            ……
            ……
          end;
   end;
end.
```



Activité N°5 : Interruption via la broche RB0



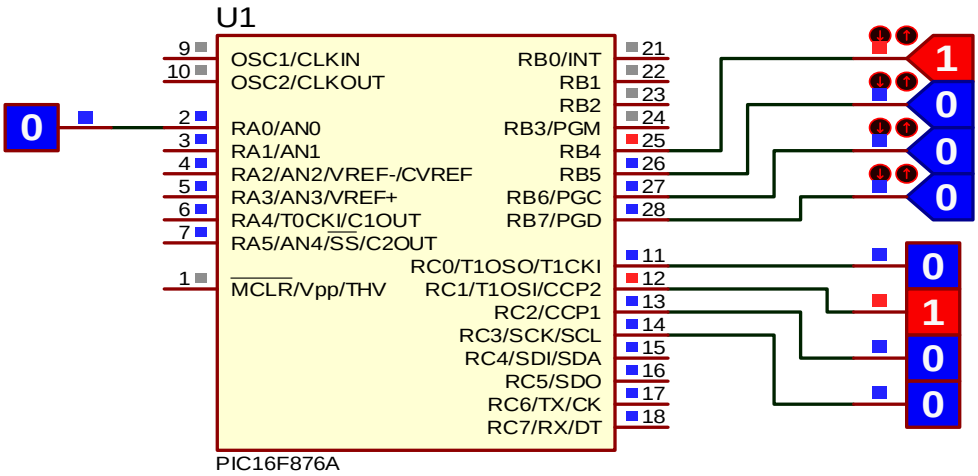
Compléter le programme suivant réalisant **un compteur modulo10 sur le PORTA**, **une interruption par RB0** activera **un décompteur modulo 10 sur le PORTC.**

```
Program ACT5;
VAR i,j: byte;
 Procedure interrupt;
 begin
  for j:= 9 ……..
      begin
        ……..
        delay_ms(500);
      end;
  ……..
 ……..
 end;
```

```
begin
trisb:=$....      trisa:=…        trisc:=0;
porta:=0; portc:=0;
intcon:=……
while true do
    begin
       for i:=0 to 9 do
          begin
             porta:=i;  ……..
          end;
    end;
end.
```

Activité N°6 : Interruption via les broches RB4, RB5, RB6 ou RB7

Compléter le programme Mikropascal réalisant **un chenillard sur le PORTC (RC0, RC1, RC2, RC3, RC4),** le changement d'état d'au moins un des capteurs déclenche **une interruption : clignotement d'une led D0 (RA0) pendant 3 secondes :**




```
Program ACT6;
Var
diodes : byte at ……
D0: sbit at ……
i,j, etat :byte;

Procedure interrupt;
begin
……..        // lecture du portB pour
             déverrouiller l'accès au bit RBIF
for j:=0 to 3 do
begin
……..
D0 := 0 ; Delay_ms(500);
end;
……..  // remise à zéro du drapeau RBIF
INTCON.GIE :=1 ; // ……..
end;
```

```
begin
trisb:=%.....
trisa:=……..
trisc:=$.....
porta:=0 ;
portc:=0 ;
…….. //  validation de l'interruption RBI
while true do
Begin
diodes := %1000;
For i :=0 to 3 do        // Compteur de 0 à 3
begin
……..  // pause de 0,5s
…….. // décalage à droite d'un pas
end;
end;
end.
```

Activité N°7 : Compteur modulo10 en utilisant le timer TMR0

On désire réaliser un compteur modulo10 en utilisant le timer TMR0.

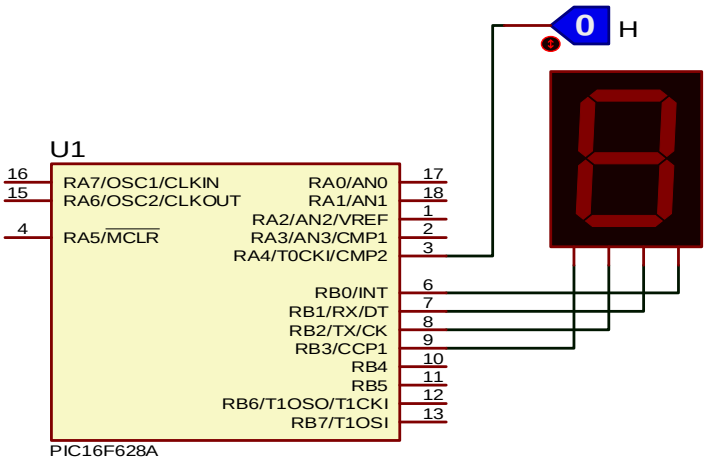
« Entrée de comptage sur RA4 »↑ avec un coefficient de prédivision =1 (sortie sur le port B)

1) Configurer le registre **« OPTION_REG »**

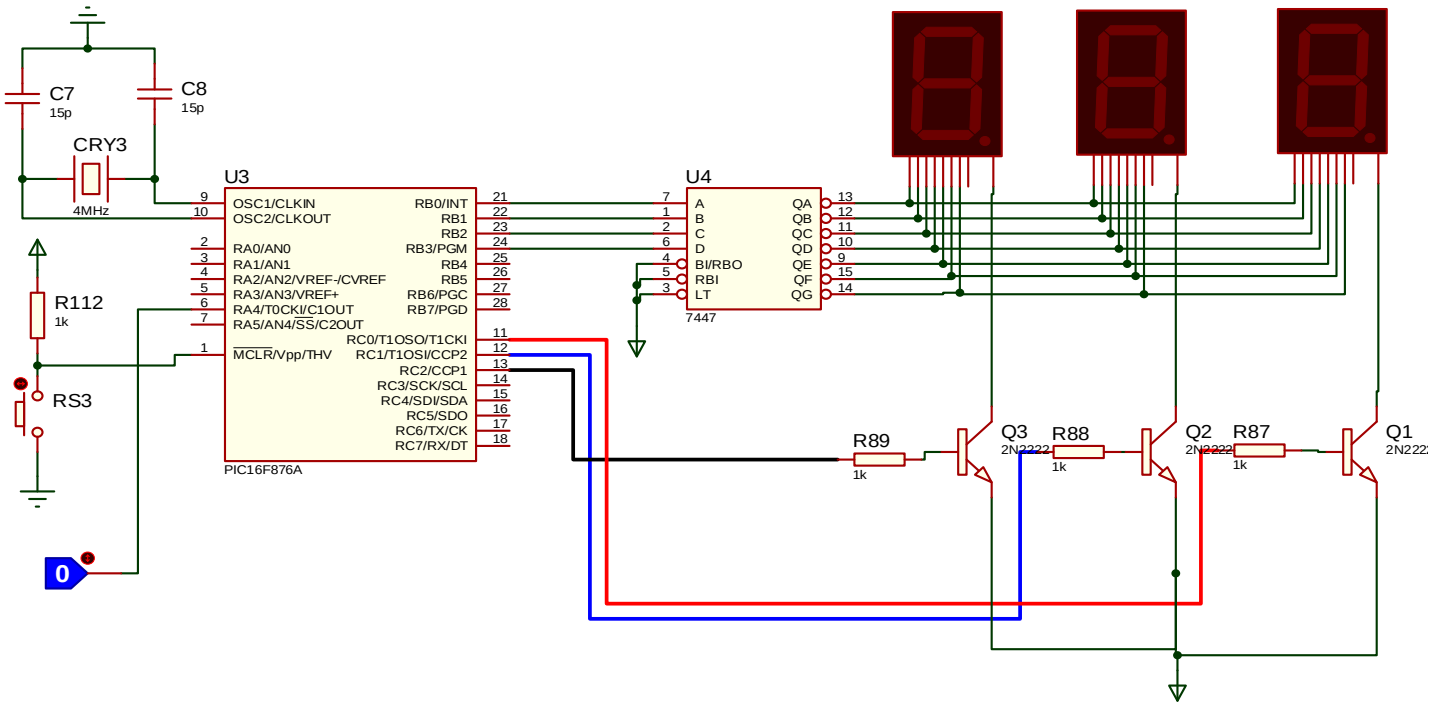
RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

2) Compléter puis saisir le programme ci-contre puis simuler et vérifier le fonctionnement :

```
Program ACT7;
begin
   TRISB:=$.....
   TRISA:=$.....
   OPTION_reg:= %.....
   TMR0:=0;
   while (1=1) do
   begin
     portb:=TMR0;
     ……..
    end;
end.
```

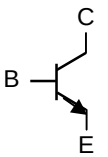


Activité N°8 : Compteur modulo 300 en utilisant le timer TMR0 et avec affichage multiplexé



Le principe est de placer le nombre à afficher sur le décodeur puis commander le transistor correspondant pour l'afficher.

Les transistors utilisés pour la commande des afficheurs sont de type **NPN**.



- Si I_B=1 le transistor est saturé

- Si I_B =0 le transistor est bloqué

```
Program ACT8;
 Var    i: word;      a, uni,dix,cent : byte;
const          aucun_afficheur : byte = %000;
               afficheur1 : byte = %.....          // commande de l'afficheur 1 (unités)
               afficheur2 : byte = %.....          // commande de l'afficheur 2 (dizaines)
               afficheur3 : byte = %.....          // commande de l'afficheur 3 (centaines)
begin
trisB:=$....       trisC:=$....       trisA:=$.....
i:=0;a:=0;                                  //  Initialisation des variables i et a  à la valeur 0
……..                                       // initialisation du timer 0 à la valeur 0
OPTION_REG := %.....   // TMR0 en mode compteur qui s'incrémente toutes les deux impulsions.
while true do
begin
               if TMR0 > 200 then
               begin
               TMR0:=TMR0 – 200 ;          //   pour étendre le comptage à des valeurs supérieures à 255
               a:=a + 1 ;
               end;
               i:=TMR0 + a*200 ;
     uni  := ……..
     dix  := ……..
     cent := ……..

     For j: = 1 to 15 do
          begin
          portC:=aucun_afficheur; delay_ms(1);
          portb:=uni; portC:=afficheur1; delay_ms(10);
          portC:=aucun_afficheur ; delay_ms(1);
          ……
          portC:=aucun_afficheur ; delay_ms(1);
          ……
          end ;
    if …….. then Begin tmr0:=0; a:= 0 ; end;
end ;
end.
```

Comment identifier les unités, les dizaines, les centaines d'un nombre ?

N=3285(10)		
Unité de N	5	N mod 10
Dizaine de N	8	(N div 10) mod 10
Centaine de N	2	(N div 100) mod 10
Millier de N	3	N div 1000

Calcul du temps d'affichage Tf

$Tf = [(10^{-3} + 10x10^{-3})x3]x15$

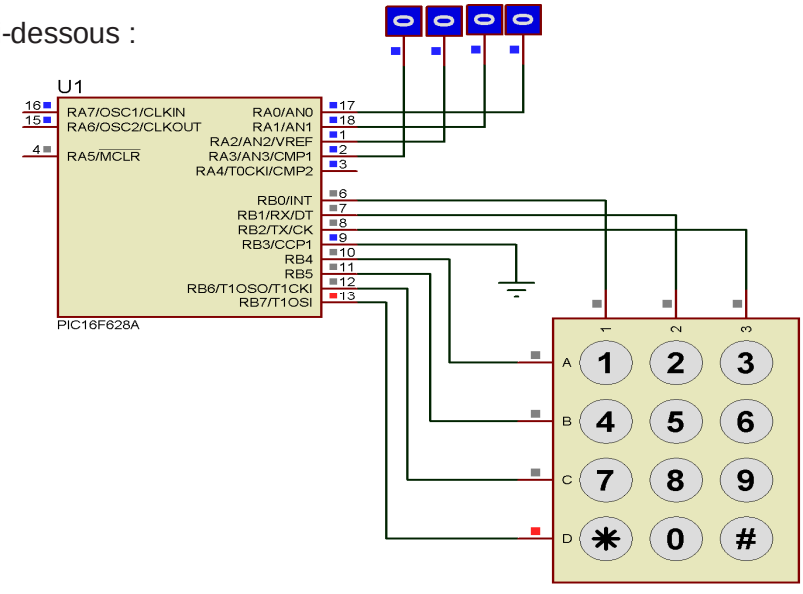
Aucun aff Afficheur i 3Affi j = 15

Tf = 0,5 seconde

Activité N°9 : Gestion d'un clavier

Soit le montage et le programme donnés ci-dessous :

```
Program ACT9;
var keypadPort : byte at PORTB;
var kp : byte;
begin
trisb:=$FF; trisa:=0; porta:=0;
CMCON:=$07;
Keypad_Init();
while true do
  begin
   kp := Keypad_Key_Click();
   if kp <> 0 then porta:=kp;
  end;
end.
```



Simuler le fonctionnement et déduire le rôle de ce programme :

Ce programme permet ……

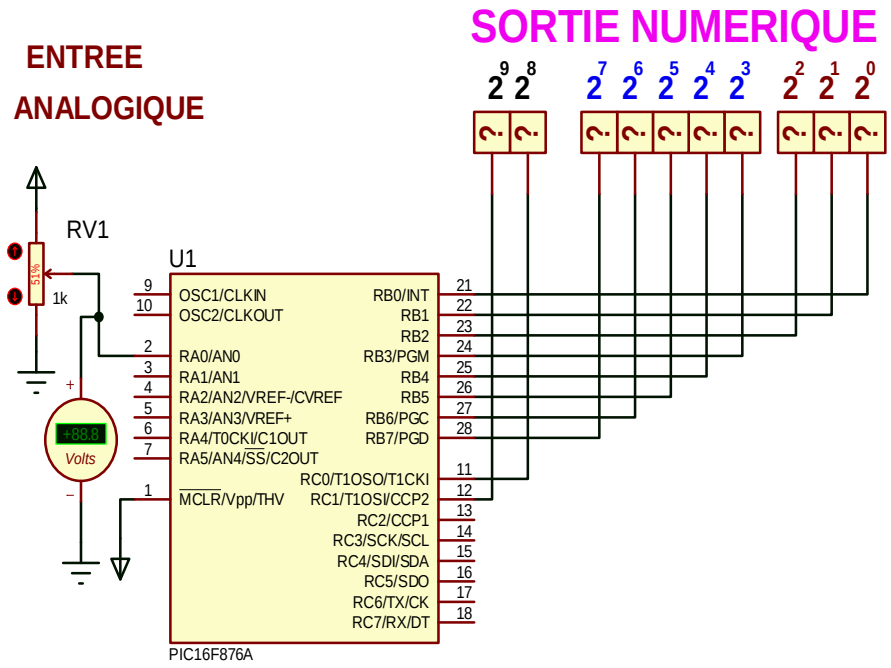
Pour chaque touche appuyée, retrouver le code correspondant en décimal. Compléter la table suivante :

Touche	1	2	3	4	5	6	7	8	9	*	0	#
Code en décimal												

Activité N°10 : Conversion Analogique Numérique : CAN ou ADC

EXEMPLE N°1 :

On désire convertir une tension variable en un mot binaire de 10bits variable.



Program ACT10EXP1;

Var **VN : ……** // VN est sur 16 bits

begin

ADCON1:=% ……

// Justification des 10 bits à droite

// et RA0 entrée Analogique (exemple)

TRISA := …….

TRISB := …….

TRISC: = …….

While TRUE do

begin

VN := ADC_Read(0); // ……

…… // Les 8 bits de plus faibles poids sont aux PORTB

PORTC := VN shr 8; // Afficher les 2 bits de fort poids dans RC0 et RC1

end;

end.

EXEMPLE N°2 : Affichage de température

Pour lire et afficher la température on utilise un microcontrôleur **PIC16F876A**, un afficheur **LCD 2*16** et un capteur de température **LM35** qui fournit à sa sortie une tension proportionnelle à température **(10mV/°C)** :

1- Exprimer **T** en fonction de **Uc** :

T ---> Uc

1°C ---> 10mV

T = f(Uc)

T = ……

2- Exprimer **Uc** en fonction de **N**, en déduire **T** en fonction de **N**:

Uc ---> N

5V ---> 1023

Uc = f(N)

Uc =……

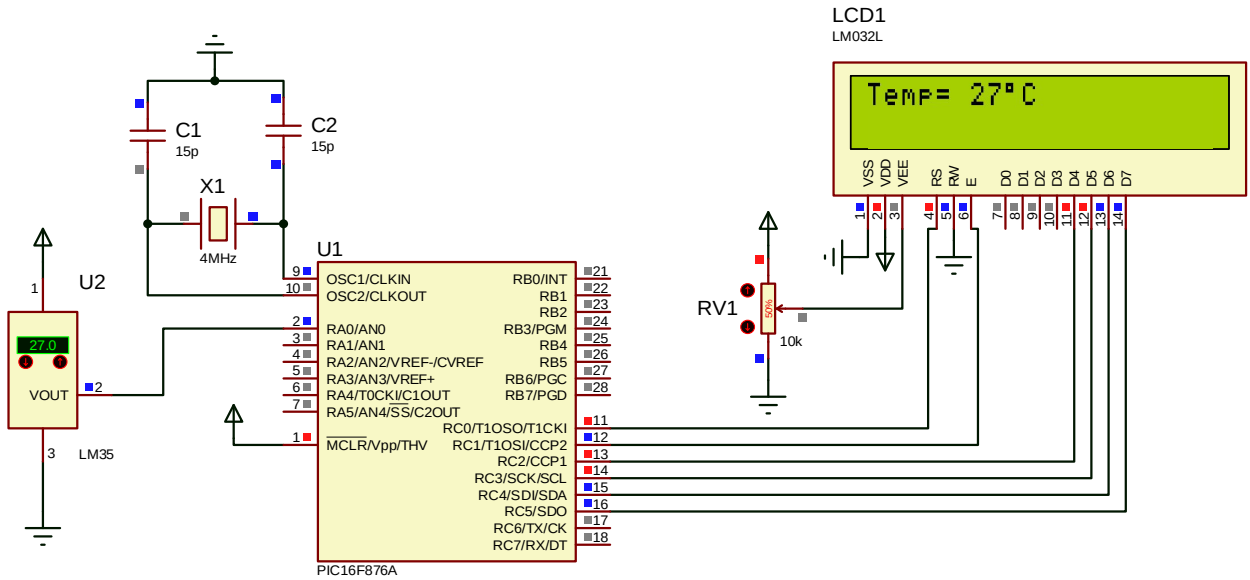
T = f(N) = ……

Programmation en langage Mikropascal :

Programme ACT10EXP2 ;	
Var Ncon : ……..	// déclaration des variables // 2 octets car le résultat de conversion est sur 10 bits
Ncal : real	// 4 octets pour le calcul
temp : ……..	// 1 octet car la température est comprise entre 2 et 150
temp_aff: ……..	//chaine de 3 caractères pour afficher la température
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;	// Connections de l'LCD

Begin
……..
lcd_cmd(_LCD_CURSOR_OFF);
lcd_out(1,1,'Temp=');
……..
	// initialisation de l'LCD
	// ……
	// préparation de l'affichage, afficher « Temp= »
	// initialisation du module CAN

while true do
begin
……
……..
……
……..
lcd_out(1,6,temp_aff);
lcd_chr(1,9,%11011111);
lcd_chr(1,10,'C');
delay_ms(100);
end;
end.
	// boucle infinie
	
	// convertir la tension Uc au RA0 en un nombre Ncon
	// conversion Ncon en Ncal (réel)
	// conversion : Ncal (réel) ---> Temp (Byte)
	// conversion : Temp (byte) ---> Temp_aff (caractère)
	// affichage de la contenue de Temp_aff
	// affichage du symbole degré: ° « voir code ASC II»
	// affichage de C pour Celsius
	……



Activité N°11 : Modulation de largeur d'une impulsion : MLI ou PWM

1- Principe :

La PWM est un module intégré dans le PIC qui consiste à générer un signal carré de période constante mais à rapport cyclique α variable. **(C'est le principe d'un hacheur série)**

N ----------> apha
1023 ----------> 255

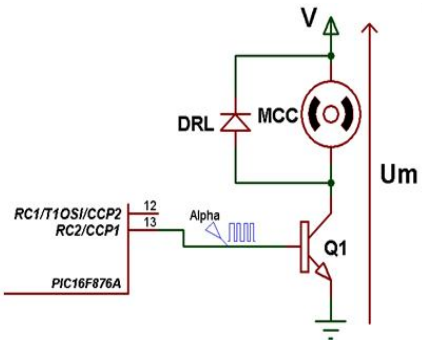
apha = (255/1023)*N

- **PIC16876A et PICF877A : possèdent**
- **deux modules PWM : CCP1 et CCP2**

- **RC1 : Sortie module CCP2**
- **RC2 : Sortie module CCP1**

aplha = 255*α

<Um> = α*V



2- Programmation en langage Mikropascal :

Programme	Commentaires

program ACT11;
var
alpha : byte;
N : word;
begin
……
……
PWM1_start;
while true do
begin
……
alpha:=N/4;
……
end;
end.
	
	// déclaration variable rapport cyclique alpha
	// nombre numérique (0 à 1023)
	// début programme
	// initialiser le MLI à une fréquence 250Hz
	// configuration du registre ADCON1
	// …….
	
	// convertir tension au canal RA0 en nombre N
	// **alpha =Nx255/1023 = N/4**
	// Rapport cyclique
	
	// ……..