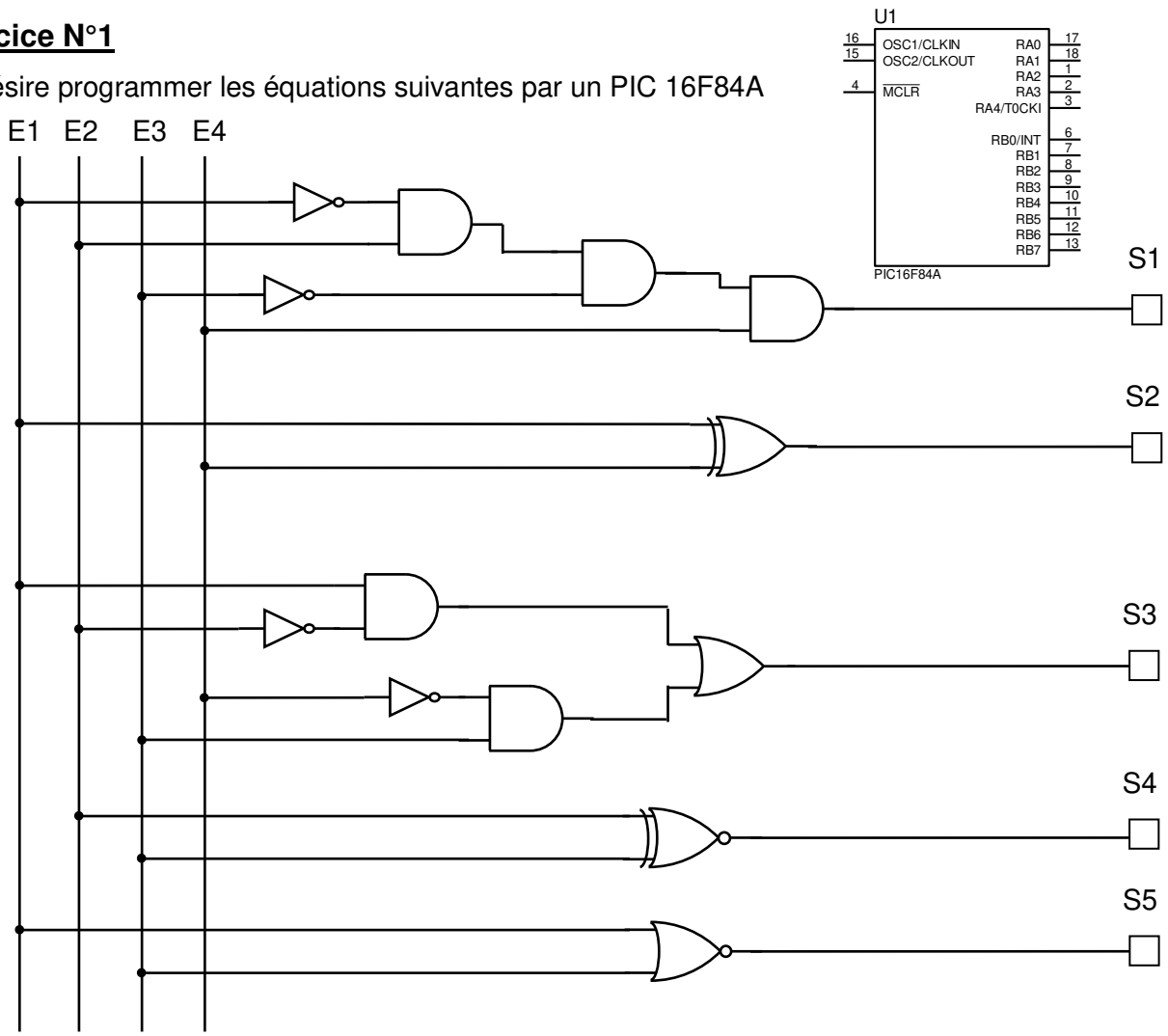
Exercice N°1

On désire programmer les équations suivantes par un PIC 16F84A



On donne le tableau d'affectation des entrées et de sorties :

Affectation des entrées		**Affectation des sorties**	
Entrées	Entrées PIC	Sorties	Sorties PIC
E1	RA0	S1	RB0
E2	RA1	S2	RB1
E3	RA2	S3	RB2
E4	RA3	S4	RB3
		S5	RB4

Opérateurs logiques réalisés par le pic

Opérateur	Opération
NOT	NON
OR	OU
AND	ET
XOR	OU exclusif

1°) compléter le tableau suivant par : Entrée « E », sortie « S », Non connecté « NC » :

PORT A		PORT B	
RA0		RB0	
RA1		RB1	
RA2		RB2	
RA3		RB3	
RA4		RB4	
		RB5	
		RB6	
		RB7	

2°) Trouver les valeurs qu'on devra placer dans les registres TRISA et TRISB du PIC16F84A :

TRIS A									= (.....)Hex
TRIS B									= (.....)Hex

3°) Etablir les équations des sorties :

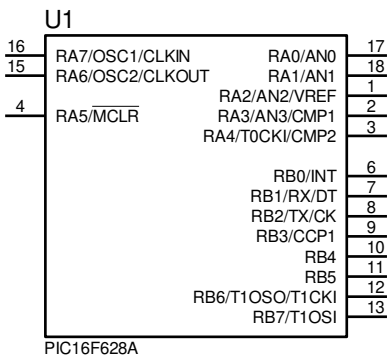
S_1 = …… S_2 = ……

S_3 = …… S_4 = ……

S_5 = ……

4°) Compléter le programme en MikroPascal correspondant aux équations précédentes :

```
program exercice1_equation;
  var
E1:Sbit at RA0_bit;
E2:…….. ;
E3:……;
E4:……;
S1:…… ;
S2:……..;
S3:……;
S4:……;
……..
Begin
Trisa:=$..... ; // tout le portA est configuré comme entrées
Trisb:=$..... ; // RB0,RB1,RB2,RB3,RB4 :sorties  RB5,RB6,RB7 : entrées
Portb :=…… ;       // initialisation
While true do
      Begin
S1:=…… ;              // equation de S1
S2:= ……  ;            // equation de S2
S3:= …… ;             // equation de S3
   ……..               // equation de S4
…….. // equation de S5
End ;
END.
```



5°) Si on veut modifier le microcontrôleur PIC 16F84A par un PIC 16F628A

Quelle est l'instruction à ajouter au programme précédent pour programmer ces équations

Exercice N° 2:

Soit le schéma à contact suivant :

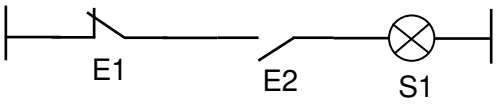
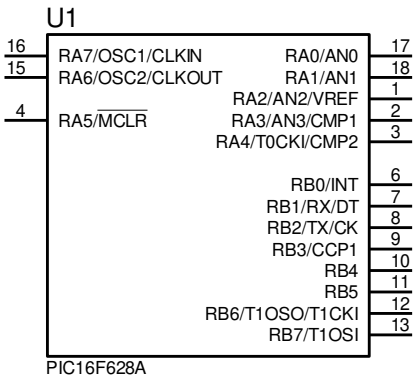


Tableau d'affectation d'entrées et de sortie

E1	E2	S1
RA0	RA1	RB0



1°) Déduire l'équation logique de S1

S1= ……..

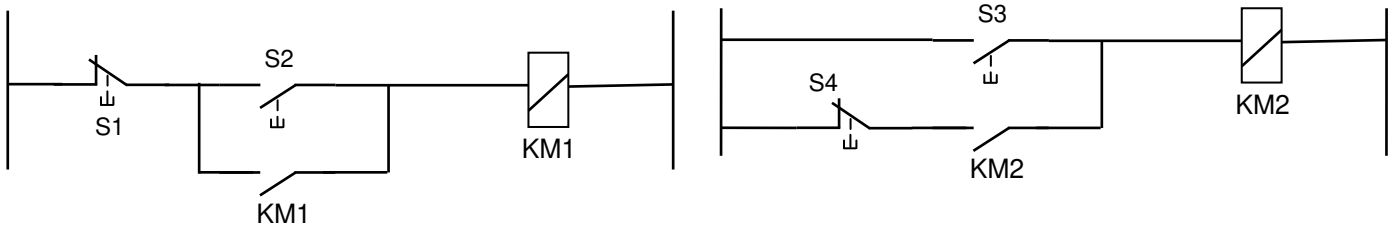
On désire programmer cette équation par un PIC 16F628A

2°) Traduire cette équation par un programme Mikropascal

```
program exercice2_equation;
Var
E1: sbit at porta.0;
..... ;
.....;
  Begin
Trisa:=$..... ;
Trisb:=$..... ;
 ……..;  // désactivation des comparateurs analogique et PORTA numérique
…… ;        // initialisation
While …… do   // boucle infinie
    Begin
  ……..    ;
  ……
 ……..
```

Exercice N° 3:

Soit les schémas à contacts suivants :



Affectation des entrées		Affectation des sorties	
Entrées	Entrées PIC	Sorties	Sorties PIC
S1	RA0	KM1	RB0
S2	RA1	KM2	RB1
S3	RA2		
S4	RA3		

On attribue à KM1 une variable de type bit (booléen) X et à KM2 une variable de type bit Y

1°) Donner l'équation de X X= ……..

2°) Donner l'équation de Y Y= ……..

3°) Traduire ces deux équations par un programme Mikropascal

```
program exercice_3_fonctions_memoires;
var
S1: sbit at RA0_bit;
S2: .....;
S3: .....
S4: .....
 X,Y:.....;
    Begin
      X:=0;Y:=0;
         Trisa:=$..... ;
         Trisb:=$.....  ;
         Portb :=.....;  // initialisation
  while true do
     Begin
        if ((S1=0) and ((S2=....) or  (X=...))) then X:=1  else X:=0;
       if ..... ;
       if X=0 then portb.0:=0 else portb.0:=...;
       if Y=0 then  .....
    .....
end.
```

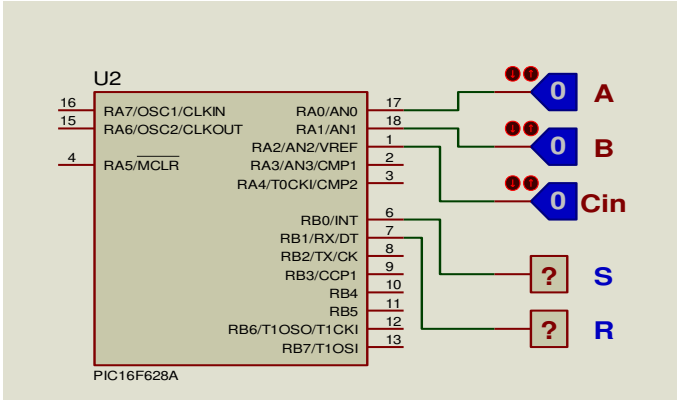
Exercice N° 4: (Additionneur complet)

On désire réaliser un additionneur complet avec le circuit 16F628A

Le circuit possède 3entrées et deux sorties :

$S = (A \oplus B) \oplus Cin$;

$R1 = (A \oplus B).Cin + A.B$

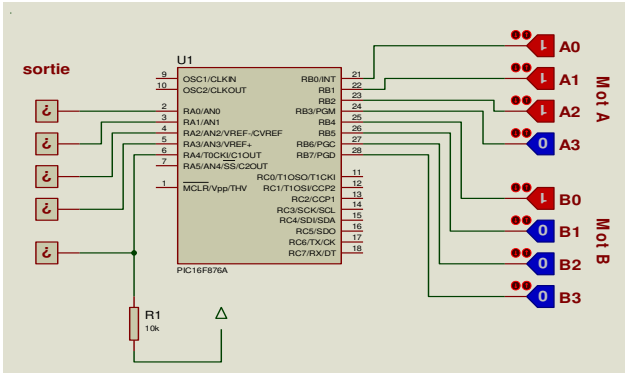


Compléter puis **s**aisir le programme ci-contre pour réaliser ce circuit :

```
program Additonneur;
  …….
A:   sbit at PORTA.0;  // variable de type bit affecté au PORTA.0 çad "RA0"
B:   ……..;
Ci:…….;
S:   ……;
R:   ……..;
begin
  TRISA:=$.....;
  TRISB:=$..... ;
…….=$07;  // Désactivation des comparateurs " PORTA numérique "
  PORTB:=……; // initialisation des sorties
  while true do begin
  …….. // équation de la somme
  …… // equation du retenu
    ……
…….
```

Exercice N° 5: (Additionneur BCD)

On désire réaliser un additionneur BCD avec le circuit 16F876A ; compléter alors le programme ci-contre:

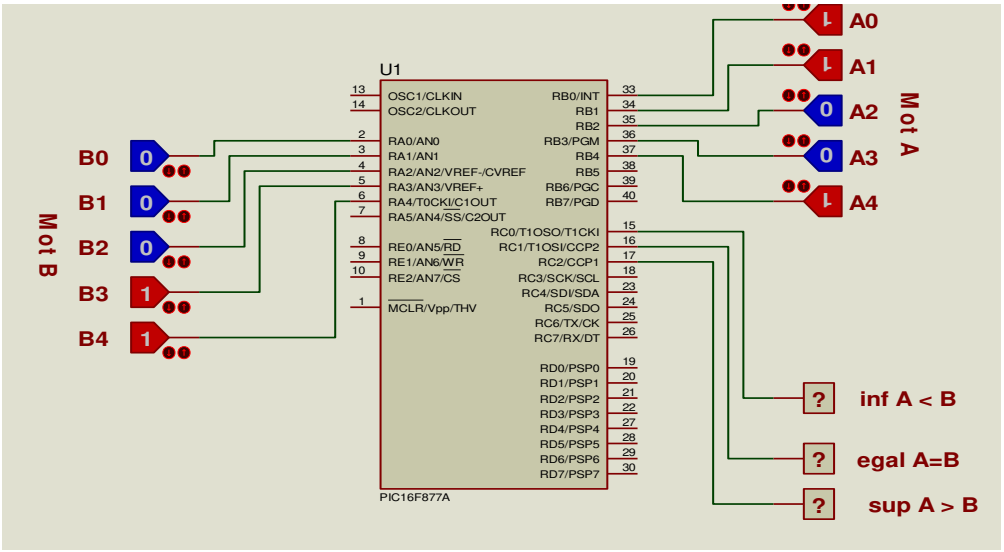



```
program additionneur_bcd;
var
A,B,S1: byte;
begin
  TRISA:=$.....;
  TRISB:=$.....;
   PORTA:=….;
    while 1=1 do
    begin
A:=PORTB;  //  La variable A  reçoit  le contenu du PORTB
 A:= A and $.....;  //  Masquer les 4 bits de poids le plus fort
    B:=PORTB;  //  La variable A  reçoit  le contenu du PORTB
    B:=B shr ……. ;  //  Décalage à droite de 4 bits
    S1:= ……. ;  //  Addition de A et B
    if S1 > …… then S1:= S1 + ……. ;  // Correction  si  la somme est > 9
    PORTA:= ……..;
…….
end.
```

Exercice N° 6: (comparateur)

On desire réaliser un comparateur de deux nombres de cinq bits avec le PIC 16F877A

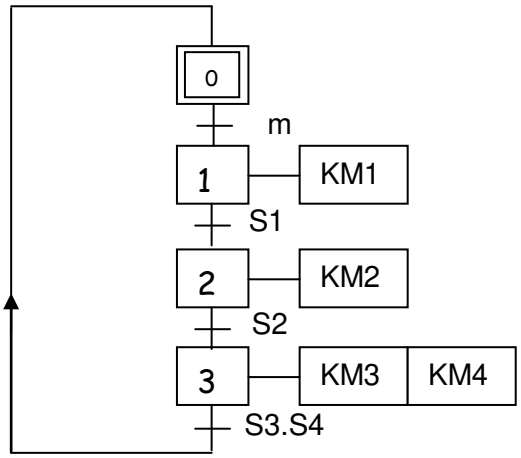
compléter alors le programme ci-contre:

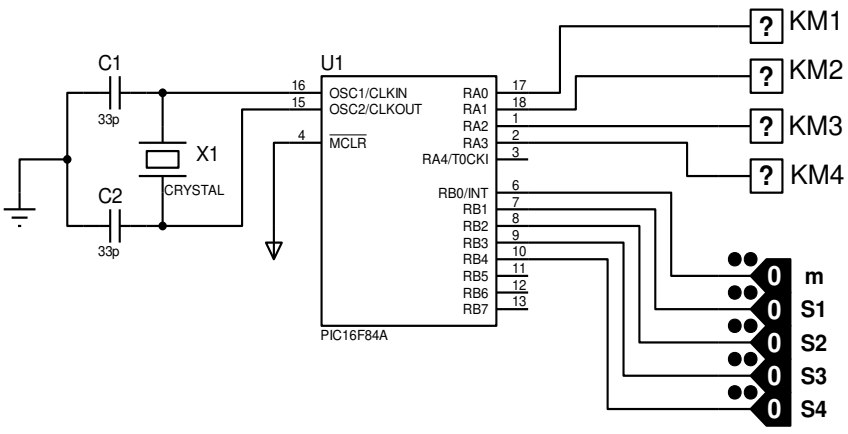



```
program comparateur;
var
A:byte at portB;
B:……;
inf:……;
ega:……;
sup:……;
begin
  TRISA:=$.....;
  TRISB:=$.....;
  TRISC:=$.....;
  PORTC:=…;
  ADCON1:=$.....; // tout le port A est numérique
  while true do
  begin
   if A < B then inf:=…. else inf:=…….;
   ……..
……..
    end;
……
```

Exercice N°7:(GRAFCET1)

Le fonctionnement d'un système est décrit par le GRAFCET suivant:

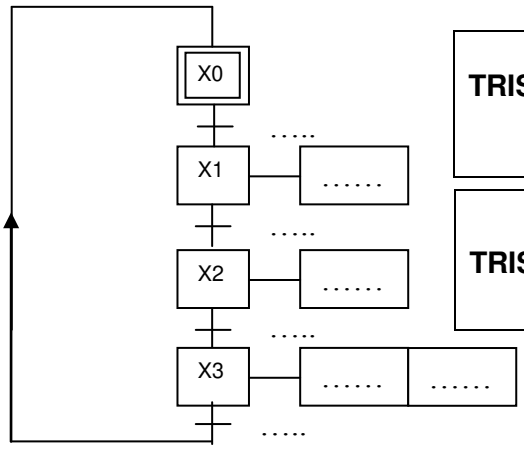




Circuit de commande du système

1°) Compléter le GRAFCET codé microcontrôleur

2°) Compléter les affectations des deux registres **TRIS A** et **TRIS B**.



TRIS A				RA4	RA3	RA2	RA1	RA0
				……	……	……	……	……

TRIS B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	……	……	……	……	……	……	……	……

3°) compléter le programme relatif au grafcet précédent

```
program exercice7_grafcet1;
    Var
  m: sbit at RB0_bit ;
  ……..
  ……..
  ……..
  ……..
  ……..
  ……
   …….
……..
X0,X1,X2,X3:……;
        BEGIN
          ……..... ;          // Configuration du registre TRISA en Hexadécimal
          ……..... ;          // Configuration du registre TRISB en Hexadécimal
          …….. …….. ;        // Initialisation du portA
 X0:=1 ; X1:=0 ;…… ; // Initialisation des étapes du grafcet
              ..... // boucle infinie
          BEGIN
 IF ((……) AND (……)) THEN          // Condition d'activation de l'étape1
                    BEGIN
                      ..... ;
                    END ;
 IF ((……) AND (……..)) THEN         // Condition d'activation de l'étape2
                    BEGIN
                    X1 := 0 ; X2 := 1 ;
                   END ;
```

…… **// Condition d'activation de l'étape3**

BEGIN

.....

END ;

.... **// Condition d'activation de l'étape0**

……

.....

.....

IF (X1=1) THEN KM1:= 1 ELSE …… // **Programmation de la sortie KM1**

.....// **Programmation de la sortie KM2**

.....// **Programmation de la sortie KM3**

..... //**Programmation de la sortie KM4**

……..;

END.

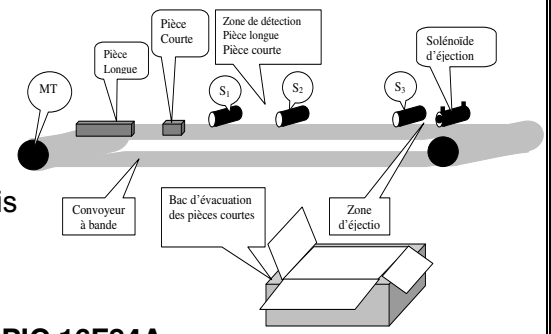
Exercice N°8:(GRAFCET2)

Système : chaîne fonctionnelle :

On peut assimiler la chaîne fonctionnelle à un système de tri de pièces.

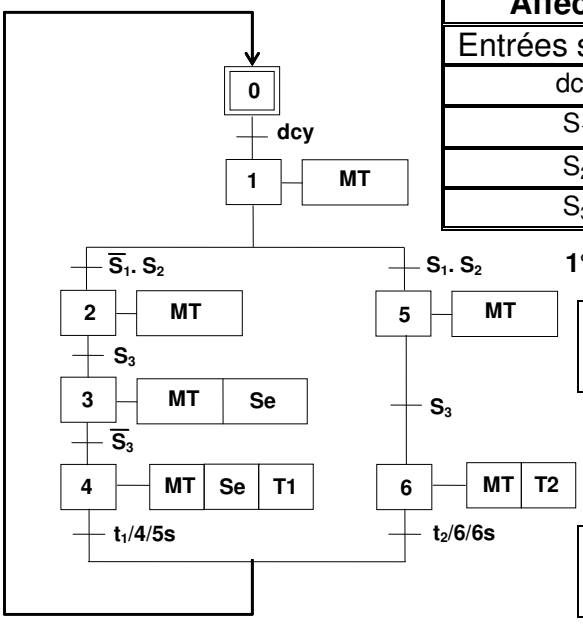
Les pièces longues et les pièces courtes arrivent sur le même convoyeur :

- Si la pièce est longue, elle doit passer jusqu'à la fin du tapis pour être évacuée.
- Si la pièce est courte, elle doit être éjectée dans un bac.



On veut commander la chaîne fonctionnelle par un microcontrôleur **PIC 16F84A.**
On donne le grafcet P.O et le tableau d'affectation des entrées et sorties

GRAFCET d'un point de vue P.O



Affectation des entrées		Affectation des sorties	
Entrées système	Entrées PIC	Sorties système	Sorties PIC
dcy	RA0	MT	RB0
S_1	RA1	S_e	RB1
S_2	RA2		
S_3	RA3		

1°) Compléter les affectations des deux registres **TRISA** et **TRISB**.

TRISA				**RA4**	**RA3**	**RA2**	**RA1**	**RA0**
				..	….	…	…	….

TRISB	**RB7**	**RB6**	**RB5**	**RB4**	**RB3**	**RB2**	**RB1**	**RB0**
				..	….	…	…	….

2°) Compléter le programme en langage pascal relatif à la commande du système :

```
program Exercice8_grafcet2;
Var
dcy:sbit at porta.0;
S1:…………………….
S2: …………………….
S3: …………………….
MT: …………………….
Se:sbit at portb.1;
  var  X0, X1, X2, X3, X4,X5,X6,T1,T2:…….;
    begin

    trisa := $…….; //  configuration  du portA en entrée
    trisb := $…….; portB := …….; // configuration et initialisation du portB
    X0:= ….; X1:= …. ;X2:= ….;X3:=0; X4:=0;X5:=0;X6:=0;
         ………………. // boucle infinie
         begin
         if ((X0=1) and (dcy=1)) then
                  begin
                       X0 := …….; X1 := …….;
                  end;
………………………………………………….
                 begin
                   X1 := 0; X2 :=1;
                 end;
      if((X2=…….)  and ( S3=1) ) then
                 …….
                 ……………………….
               …….
          if ((X3=…….) and ( S3 =…….)) then
                begin
         X3 := 0; X4 :=1;
              end;

         ………………………………………………….
                 begin
                   X1 := …….; X5 := …….;
                 end;
       if((X5=1)  and (S3=1))   then
                begin
                   …………………………………….
                end;
          if  ((X6=…….) and (t2=…….) ……. (X4=…….) and (t1=…….))  then
                     …….
                 X6 := …….; X4 := …….;X0:= …….
                   …….
      // programmation des sorties
        if((X1=1) ……. (X2=1) ……. (X3=1) ……. (X4=1) ……. (X5=1) or (X6=1))  then MT:=1 else MT:=0;
        ………………………………………….then Se:=1 else Se:= …….;
        // programmation des temporisations
  if (X4=1)  then  T1 := 1 else T1:=0;
  if T1=1 then delay_ms(5000);
  ………………………………………………….
  ………………………………………………….
          …….
      end.
```

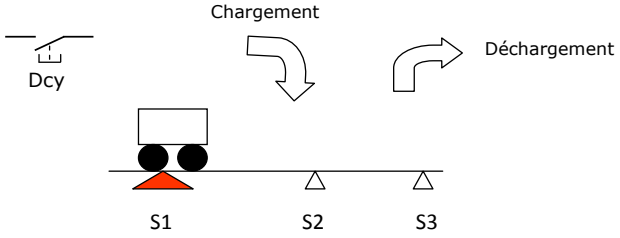
Exercice N°9:(GRAFCET3)

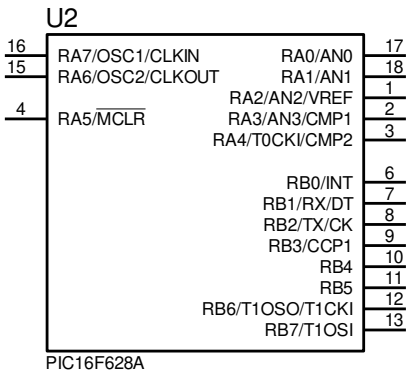
Système : déplacement d'un chariot:

Le chariot étant en position initiale (S1 est actionné) un ordre de départ cycle « dcy » provoque :

- Le déplacement du chariot jusqu'à « S2 »
- Le chargement du chariot avec un produit et une temporisation de 10s.
- Le déchargement en « S3 »
- Retour du chariot en « S2 » pour le charger et le décharger de nouveau en « S3». Enfin, il revient en « S1 ».

Un compteur est incrémenté à la fin de chargement, sa sortie n=1 si le chariot est chargé 2 fois.



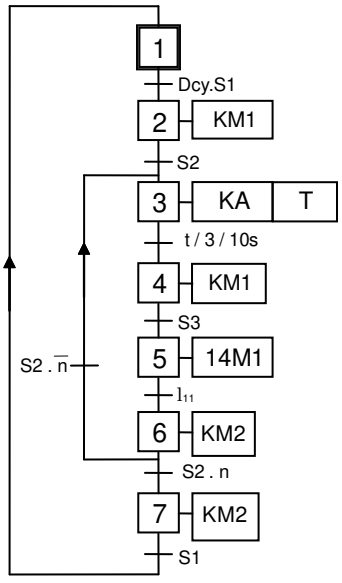


On veut commander le système par un microcontrôleur **PIC 16F628A.**

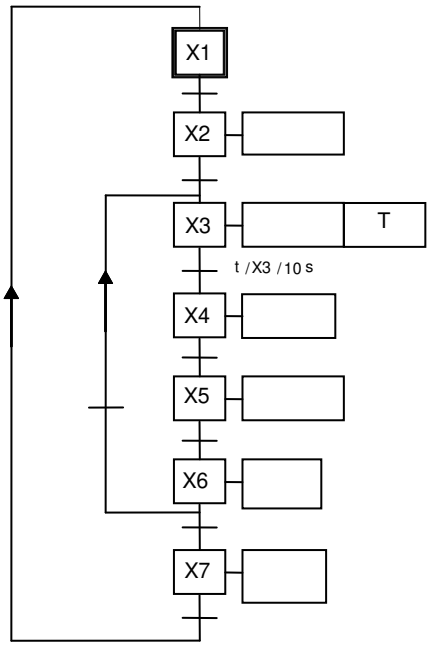
En se référant au grafcet P.C et au tableau d'affectation des entrées et sorties

1°) Compléter le GRAFCET codé PIC

GRAFCET P.C



GRAFCET codé PIC



Affectation des entrées		Affectation des sorties	
Entrées système	Entrées PIC	Sorties système	Sorties PIC
Dcy	RB0	KM1	RA0
S1	RB1	KM2	RA1
S2	RB2	14M1	RA2
S3	RB3	KA	RA3
I_{11}	RB5		
n	RB6		

2°) Compléter les affectations des deux registres **TRISA** et **TRISB**.

TRISA	**RA7**	**RA6**	**RA5**	**RA4**	**RA3**	**RA2**	**RA1**	**RA0**
	……	……	……	……	……	……	……	……

TRISB	**RB7**	**RB6**	**RB5**	**RB4**	**RB3**	**RB2**	**RB1**	**RB0**
	……	……	……	……	……	……	……	……

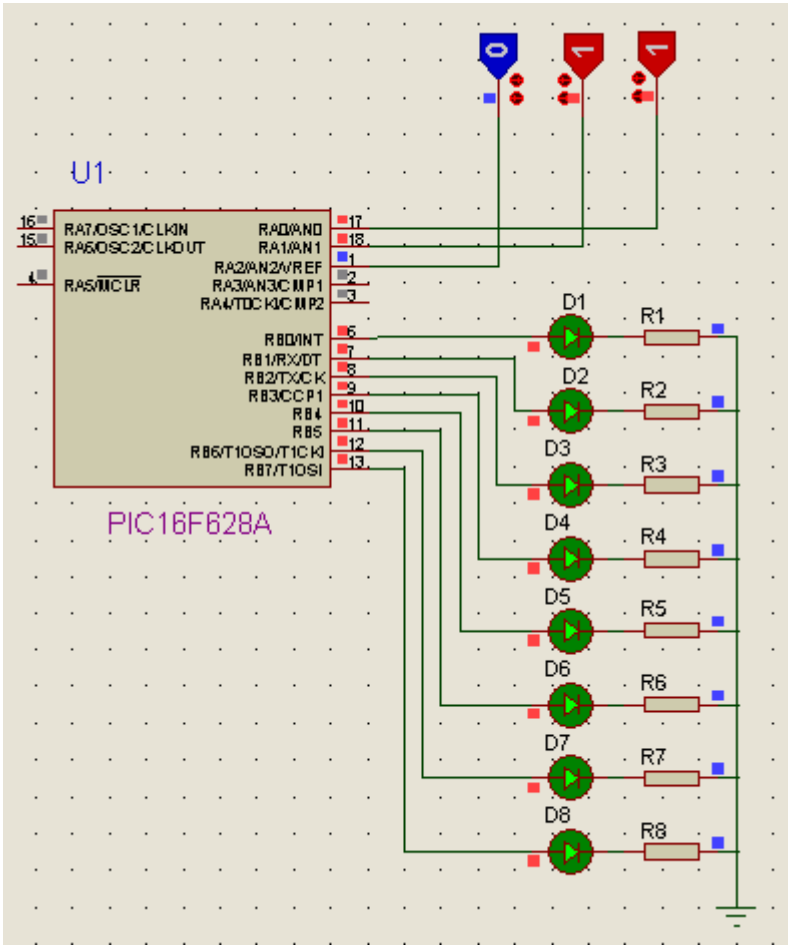
3°) compléter le programme en langage pascal relatif à la commande du système :

```
program EXERCICE_9GRAFCET3;
  …..
dcy:…….
S1:……..
S2:…….
S3:……..
L11:…….
n:……
KM1:……
KM2:……
v14M1:sbit at portA.2;
KA:……
var  X1, ……., ……, ……..,……,……..,X7,T : …….;
    begin
    CMCON:=$.....; // désactiver les comparateurs analogiques et porta numérique
trisA:=$.....; trisB:=$.....; portA:=…….;  // configuration des entrées sorties et initialisation
X1 :=……; X2:=……;X3:=…….;X4:=……;X5:=……;X6:=……;X7:=…….;
        while ..... do     // boucle infini
          begin
 if ((X1=1) and (Dcy=1) and( S1 =1)) then
                    begin
                        X1 := 0; X2 := 1;
                    end;
 if ((X2=1)  and ( .....) or (X6=1)  and (.....)  and(.....))  then
                   begin
                     X2 := 0; X6 := 0; X3 :=1;
                   end;
   if (.....and (t=1))   .....
                  begin
                  X3 := 0; X4 :=1;
                 end;
```

```
    if ((X4=1) and ( S3 =1)) then
                begin
          ..........................................................
                end;
    if ((X5=1) and (.....)) then
                begin
                 ..........................................
                end;
          if ((X6=1)  and (   .......    .) and( ..........    )) then
                  begin
                     X6 := 0; X7 :=1;
                  end;
         ..................................................
                begin
                   X7 := 0; X1 :=1;
                end;
// programmation des sorties
if(X2=1)or (X4=1) then KM1:=…. else KM1:=0….;    // programmation de la sortie KM1
……..    // programmation de la sortie KM2
if(X5=1) then v14M1:=1 else v14M1:=0;    // programmation de la sortie 14M1
……..    // programmation de la sortie KA


// programmation du temporisation
  if (X3=0)  then  t := 0 else
     begin
     delay_ms(……); t := ……;
       end;
             ……;
        end.
```

ExerciceN°10

Soit le montage suivant permettant de commander 8 diodes LED avec un microcontrôleur PIC 16F628A :



Fonctionnement :

Tableau résumant le fonctionnement des diodes

PORTA	RA2	RA1	RA0	Etat des diodes
…	0	0	0	Eteintes
1	0	0	1	Clignotent pendant 2s
….	0	1	0	Chenillard décalage à droite
	0	1	1	Allumées
….	1	0	0	Chenillard décalage à gauche
	1	0	1	Eteintes
	1	1	0	Allumées
	1	1	1	Clignotent pendant 2s

1°) Compléter le tableau précédent.

2°) Déduire les valeurs des TRISA et TRISB du microcontrôleur :

TRISA									= (.....)$_{Bin}$	= (.....)$_{HEX}$	= (.....)$_{10}$
TRISB									= (.....)$_{Bin}$	= (.....)$_{HEX}$	= (.....)$_{10}$

3°) Compléter le programme en Micropascal permettant de commander les diodes LED :

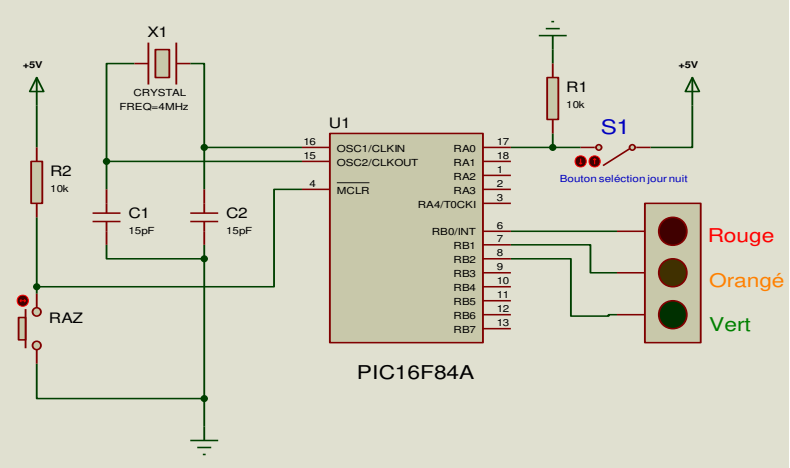
```
program exercice_10;
    var i:byte;  // declaration d'une variable i de type octet
begin
        TRISB:= ……..  ;     // Configuration du port B comme sortie
        TRISA:=$.....;  // Configuration du port A comme entrée
        PORTB:=…….;    //  initialisation du port B
        CMCON:=$.....;  //  Désactivation du comparateur, PORTA numérique
…….    //    Boucle infinie
    begin
  if  ((PORTA = 0)  or  (PORTA = …))  then PORTB:= …….;  // toutes les diodes sont éteintes
  if((PORTA = …..) ……..  (PORTA = …))    then PORTB:=$ .....;  //    toutes les diodes sont allumées
  if ((PORTA = …….)  ……  (PORTA=…….))  then              //  clignotement des diodes
                begin
                    PORTB:=$..... ;
                    delay_ms(1000) ;
                    PORTB:=$..... ;
                    …… ;
                end;
          if PORTA = ……. then       //  fonctionnement   chenillard décalage à droite
                begin
                    PORTB:=%10000000;
                        for i:=1 to 8 do    //  Boucle pour
                            begin
                                delay_ms(100);
                                PORTB:=PORTB …….1 ;   // décalage à droite de 1 bit du PORTB
                            end;
                end;
          if PORTA = ……. then       //  fonctionnement  chenillard  à gauche
                begin
                    PORTB:=%.....
                        for i:=1 to 8 do    //  Boucle pour
                            begin
                                delay_ms(100);
                                PORTB:=PORTB …….....1;  // décalage à gauche de 1 bit du PORTB
                            end;
                end;
    …….
 …….
```

ExerciceN°11 : Feux tricolores de carrefour

Tableau de fonctionnement jour :(**S1=1**)

Durée en secondes	60	05	55
Rouge (portb .0)	1	0	0
Orangé (portb.1)	0	1	0
Vert (portb.2)	0	0	1

Fonctionnement nuit :(**S1=0**) :
Le feu orangé clignote pendant une seconde.



Compléter l'algorithme et le programme donné permettant de gérer le fonctionnement du feux tricolore d'un carrefour en se référant au tableau de fonctionnement (jour et nuit) et au schéma du montage fourni.

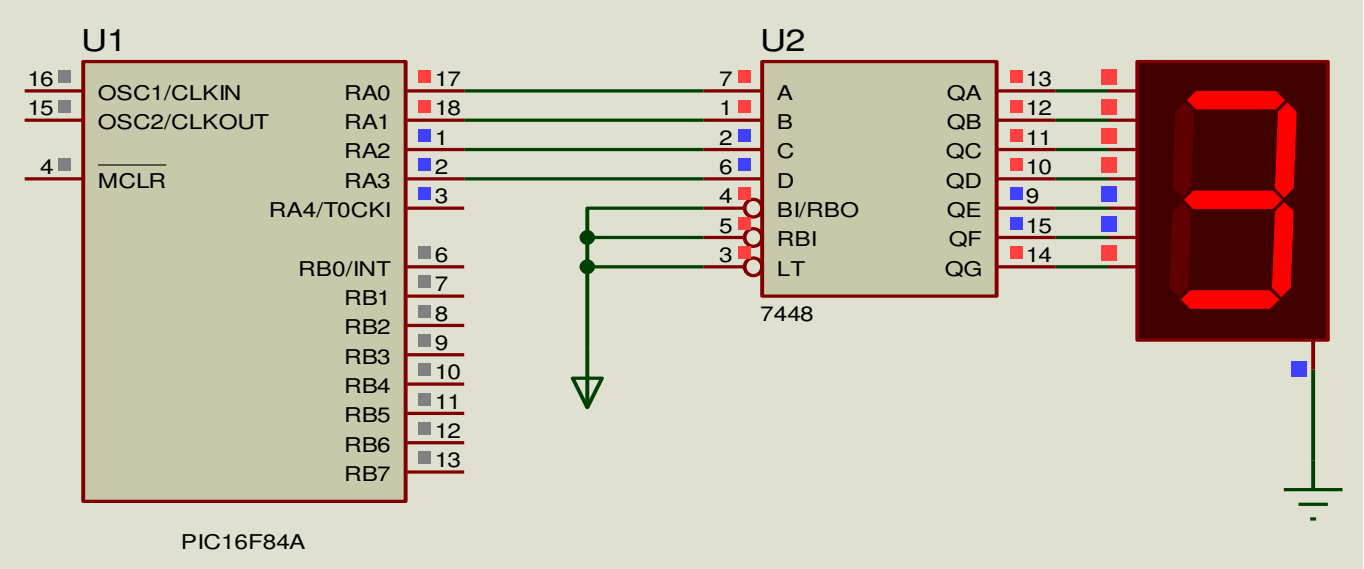
Algorithme	Programme
Algorithme : feu_tri Variables : S1 : bit du registre PORTA affecté à RA0 Début TrisA ⇐…….. TrisB ⇐…… PortB ⇐ …… Tant que (1=1) Faire Début SI S1=1 alors Début Portb ⇐ ….. Attente ( ) …… …… portb: ⇐ ….. Attente (55s) Finsi SI non Début portb: ⇐ …. Attente (0,5s) portb: ⇐ …… Attente (0,5s) Fin SI Fin Faire Fin	program feu_tri; var S1:..... begin trisA:=$ .. ; trisb:=$... ; portb:=$.....; while true do begin ..... begin ..... Vdelay_ms(60000); .... portb:=4; Vdelay_ms(55000) end ..... begin portb:=2; delay_ms(500); portb:=0; .....; end; end; end.

Exercice N°12

Compteur modulo 10

1°) Compléter l'algorithme d'un compteur modulo 10.

2°) Compléter le programme pascal correspondant.



Algorithme	Programme
Algorithme : compteur m10 Variable i :entier **Début** TrisA ⇐ % ..... PortA ⇐ % ..... Tant que vrai faire début Pour i variant de ..... jusqu'à ..... faire début PORTA ⇐ ..... Attente (0,5s) Fin faire Fin faire **Fin**	Program compteur m10 ; Var i:integer; // declaration d'une variable de type entier **Begin** TrisA := %..... ; Port ..... := %..... ; while ..... do // boucle infinie begin For i:= ..... to ..... do // boucle répétitive begin PortA := ..... ; delay_ms(.....); end; end; **End.**

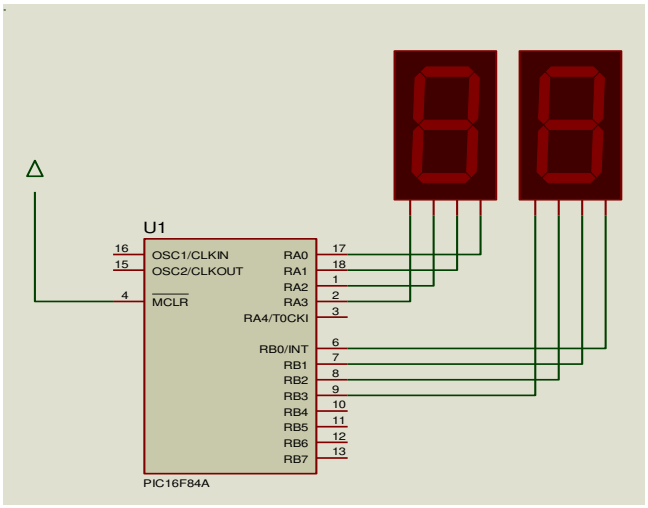
Exercice N°13

Compteur modulo 60

1°) Compléter l'algorithme d'un compteur modulo 60

2°) Compléter le programme pascal correspondant.

Algorithme	Programme
Algorithme compteur mod 60 variable i,j:entier début trisa ⇐ …. trisb ⇐ .. porta ⇐ …. portb ⇐ ….. Tant que (…….) faire début pour i variant de 0 jusqu'à 5 faire début …… début porta ⇐ … portb ⇐ … Attente (1s) finfaire fin faire finfaire fin	program compteur60; VAR i,j:integer; begin trisa:=.....;trisb:=.....;porta:=....; portb:=.... ; ..... begin ..... begin for j:=0 to 9 do begin ..... delay_ms(1000) ..... end; end; end.



Exercice N°14

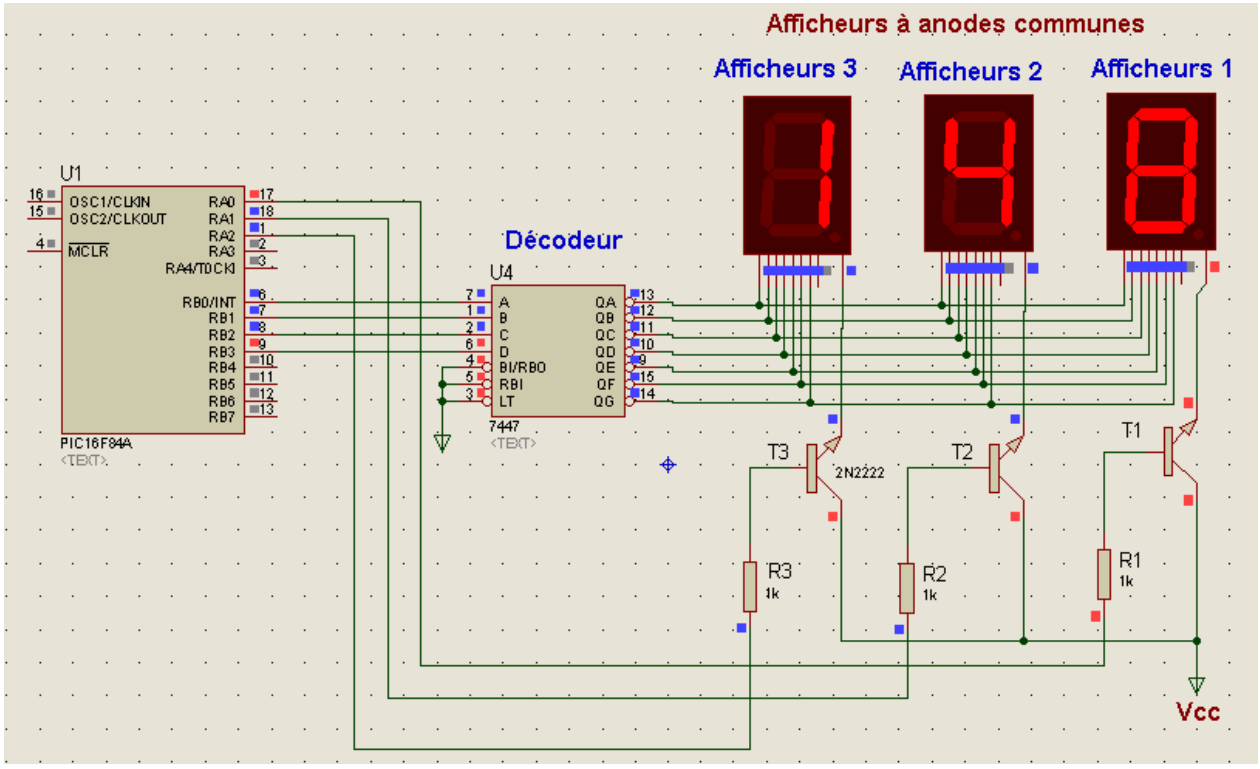
Compteur modulo 1000 avec un décodeur et 3 afficheurs 7 segments avec affichage multiplexé

Le microcontrôleur 16F84A réalise le multiplexage de l'affichage. Le décodeur utilisé est le **7447** dont les sorties sont activées à niveau bas donc les afficheurs sont à anodes communes.

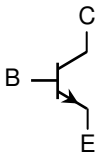
Les bornes communes des afficheurs « anodes » sont commandées à travers des transistors NPN de telle sorte que lorsque un transistor est saturé, la borne commune de l'afficheur est alors reliée au +Vcc donc l'afficheur correspondant fonctionne.

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

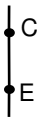
Schéma du montage



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



• Si B=1 le transistor est saturé



• Si B=0 le transistor est bloqué

Le principe est d'incrémenter une variable d'identifier son unité, son dizaine et son centaine l'envoyer au décodeur puis à chaque fois on commande le transistor correspondant

1°) Compléter le tableau suivant :

	Transistor : bloqué ou saturé			Afficheur commandé : oui ou non			Temporisation
Nombre à afficher	T1	T2	T3	Afficheur 1	Afficheur 2	Afficheur 3	
	bloqué	bloqué	bloqué	non	non	non	1ms
Unité	saturé	…	…	oui	non	non	10ms
	bloqué	bloqué	bloqué	non	non	non	1ms
dizaine	…	…	…	…	…	…	10ms
	bloqué	bloqué	bloqué	non	non	non	1ms
centaine	…	…	…	…	…	…	…

2°) Compléter le programme :

```
program exercice_N14_compteur_moduol1000_affichage_mutiplexe;
  var i:word;
  var j:byte;
  var unite,dizaine,centaine:byte;
  begin
 trisb:=$..... ; // RB0,RB1,RB2,RB3 sorties RB4 à RB7 entrées
 trisa:=$..... ; // RA0,RA1,RA2 sorties RA3,RA4 entrées
 porta:=…… ; // initialiser le PORTA
 ……..//  Boucle infinie
 begin
  for i:=0 to 999  do
  begin
   unite:= i  ……  10 ;  //  identifier le chiffre de l'unité de i
   dizaine:= (i ……10) ……10 ;   // identifier le chiffre de dizaine de i
   centaine:=  i …….. 100 ; //  identifier le chiffre de centaine de i

 for j:=1 to 28 do
 begin
 porta:=%000;
 delay_ms(1);
 portb:=unite;
 porta:=%001;    // ……..
 delay_ms(10);
 porta:=%000;
 delay_ms(1);
 portb:=……..;
 porta:=%.....;    // Commander le 2éme afficheur
 delay_ms(10);
 porta:=%000;
delay_ms(1);
 portb:=……..;
 porta:=%.....;
 delay_ms(10);
  end;
 end;
 end;
end.
```

Exercice N°15:

Réaliser un compteur et décompteur modulo10

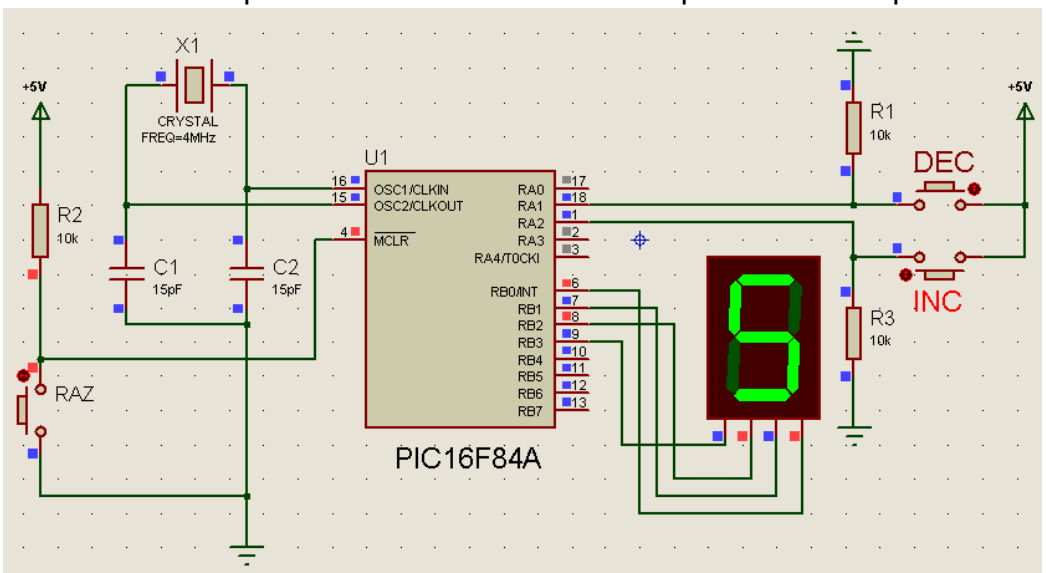
- Compteur incrémenté par le bouton RA2 .
- Décompteur décrémenté par le bouton RA1.

Utilisation de la fonction **Button** (port,bit,temps d'appui en ms ,état logique actif)

Exemple :

if Button (portA,3,50,1) then « action1 »

On teste l'appui sur un bouton poussoir relié à la broche RA3 pendant 50ms pour faire l'action 1



Compléter le programme :

```
program exercice_N_15_button;

var x:byte;

begin

 trisa:=$...;trisb:=….;portb:=….;x:=…..;

 while true do

 begin

  if button(porta,2,100,1) then ……

  if x=10 ……

   if button(porta,1,100,1) then ……

  if x=255 ……..

  portb:= ……;

  end;
  end.
```

Exercice N°16:

Réaliser un compteur modulo 10 sachant qu'a chaque impulsion sur la broche RBO le compteur s'incrémente : (Utiliser la procédure d'interruption externe avec la broche RBO)

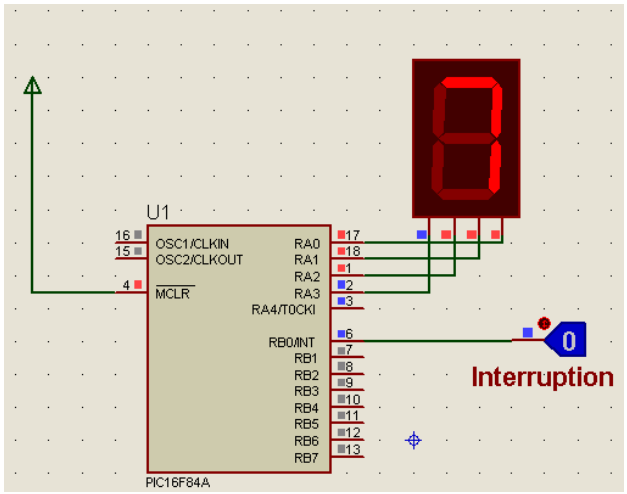
1°) Configurer le registre INTCON :

	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Valeur INTCON
INTCON									(.....)$_{16}$

2°) Compléter le programme :

Programme

```
program exercice_N_16_interrupt;
 VAR i:byte;
procedure interrupt;
  begin
  intcon :=.....;i:= .....;
  end;
   begin
 .....; intcon :=.....; i:=.....
    while(1=1) do
      begin
            .....
If  i=10  then i:=.....;

    end;
    end.
```

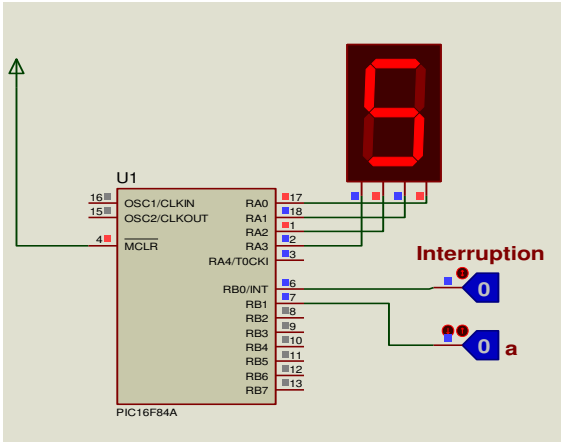


Exercice N°17

Réaliser un Compteur et Décompteur modulo 7 sachant qu'a chaque impulsion sur la broche RB0 Le C/D s'incrémente ou se décrémente suivant une entrée a :

a = 0→ compteur ; a=1 → décompteur

(Utiliser la procédure d'interruption externe avec la broche RBO) affecter la broche RB1 à l'entrée a



Compléter le programme de ce compteur et décompteur

Programme

```
program exercice_comp_decomp_modulo7_interruption;
var a:.....
var x: byte;
.....;
   begin
   if (a=1) then x:=.....
   else x:=.....
   intcon.INTF :=.....;   // .....
   intcon.GIE :=.....;       // .....
begin
intcon := $.....
trisA := $.....;
trisB := $.....;
x:=$.....;
 .....   // boucle infinie
  begin
   portA := .....;
   if  (x = $.....) then x:=$06
   else if (x = $07) then x:=$..... ;
  .....
 end.
```

Exercice N°18

Réaliser un compteur modulo 8 sachant qu'a chaque changement d'état sur au moins une des entrées RB4 à RB7 du PORTB le compteur s'incrémente : (Utiliser la procédure d'interruption externe avec RBI)

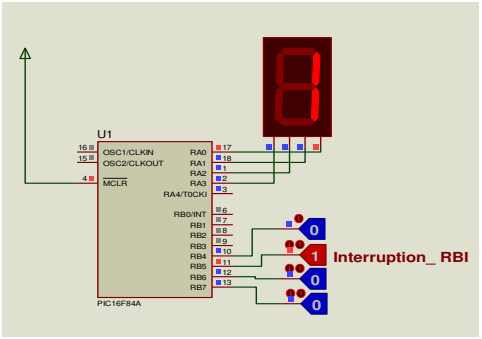
1°) Configurer le registre INTCON :

	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Valeur INTCON
INTCON									$(.....)_{16}$

2°) Compléter le programme :

Programme

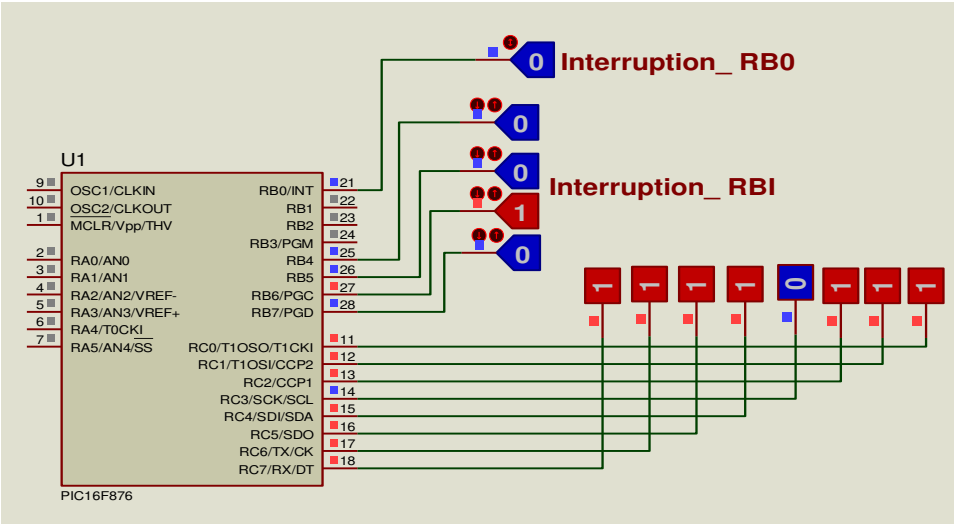
```
program exercice_N_18_interrupt;
  VAR X,etat:byte;
procedure interrupt;
  begin
  etat:=PORTB;  // lecture du portb pour déverrouiller l'accès au bit RBIF
  x:=…… ;
  intcon :=$.....;
  end;
   begin
 trisa:=…….; Trisb:=$.....;porta:=……;X:=……..;
 intcon :=$.....;
     while(1=1) do
       begin
         porta:=……;
         if ……
         end;
     end.
```



Exercice N°19

Réaliser un Compteur / Décompteur modulo 256 sachant :

- ☞ chaque impulsion sur la broche RB0 Le C/D s'incrémente: Utiliser la procédure d'interruption externe avec la broche RBO)
- ☞ chaque changement d'état sur au moins une des entrées RB4 à RB7 du PORTB le C/D se décrémente (Utiliser la procédure d'interruption externe avec RBI)



1°) Configurer le registre INTCON :

INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Valeur INTCON
									(……)$_{16}$

Programme

```
program exercice_N19_2interruption;
var
  etat:byte;
  X:byte ;
  ……..  // Procédure d'interruption
    begin
   if INTCON.intF = 1 then
  begin
……..
     intcon:=%.....
     end;
  if ……
  begin
     etat:=portb;
     ……
     intcon:=%.....
   end;
end;
begin
  TRISC:=……;
  TRISB:=$.....;
  PORTA:=…;
  intcon:=%.....;
  while true do
  begin
   ……
   end;
……
```

Exercice N°20

On désire réaliser un compteur modulo 9 en utilisant le timer TMR0. Le compteur est incrémenté à chaque front montant.

1°) Indiquer si le le mode de fonctionnement du TMR0 est compteur ou temporisateur : ……

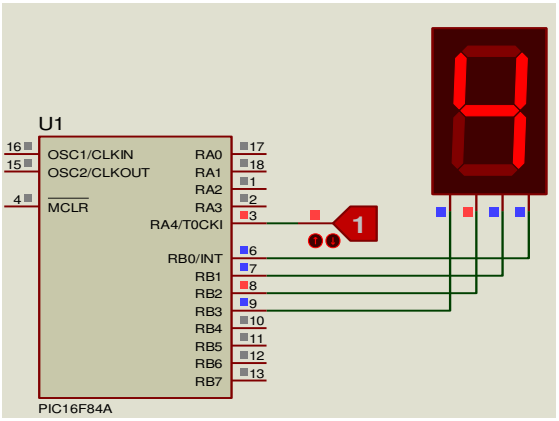
2°) Donner alors le nom de la broche de l'entrée d'horloge du TMR0 : ……

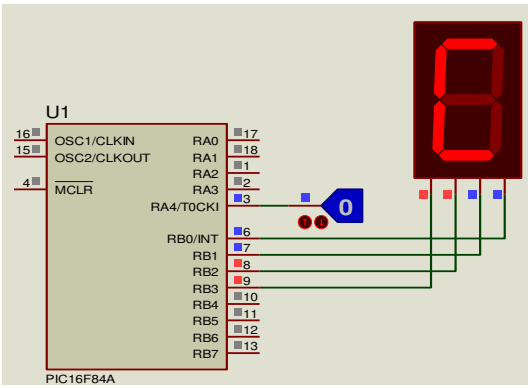
3°) Configurer alors le registre « OPTION_REG »

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PSO
1	1						

2°) Compléter le programme suivant :

```
program exercice_N_20_TIMER0_comp9;

begin

   TRISB:=$  .....;

   TRISA:=$  ..... ;

   OPTION_reg:= %.....   ;

   TMR0:=0;

   while true do

   begin

     portb:=……;

     if TMR0=……. then TMR0:=……;

     end;
end.
```





Exercice N°21

On désire réaliser un compteur modulo 16 en utilisant le timer TMR0.

Le compteur est incrémenté à chaque 2 front descendant.

1°) Configurer alors le registre « OPTION_REG »

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PSO
1	1						

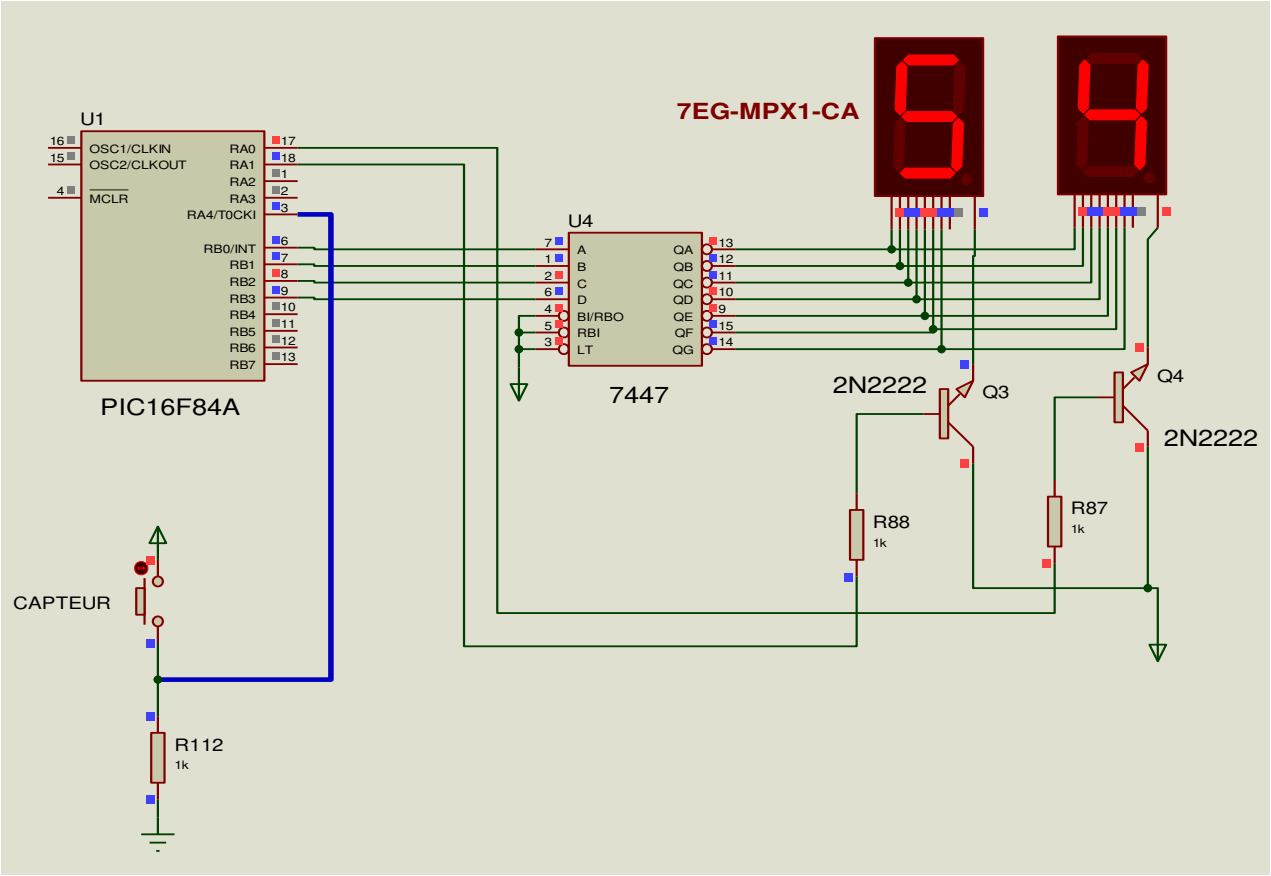
2°) Compléter le programme suivant :

```
program exercice_N_21_TIMER0_comp9;

begin

   ……..; // Configration PORTB

   …….. ; // Configration PORTA

   OPTION_reg:= %.....   ;

   TMR0:=…….; // Initialisation du TMR0

   while true do

begin

     portb:=……;

     if ……..

     ……..

end.
```

Exercice N°22

On désire réaliser un compteur modulo 100 en utilisant le timer TMR0. Le compteur est incrémenté à chaque front descendant de RA4.

On adopte l'affichage multiplexé puisqu'on dispose de d'un seule décodeur et de deux afficheurs



1°) Configurer le registre « OPTION_REG »

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	**INTEDG**	**TOCS**	**TOSE**	**PSA**	**PS2**	**PS1**	**PSO**
1	**1**				**1**	**1**	**1**

2°) Compléter le tableau suivant :

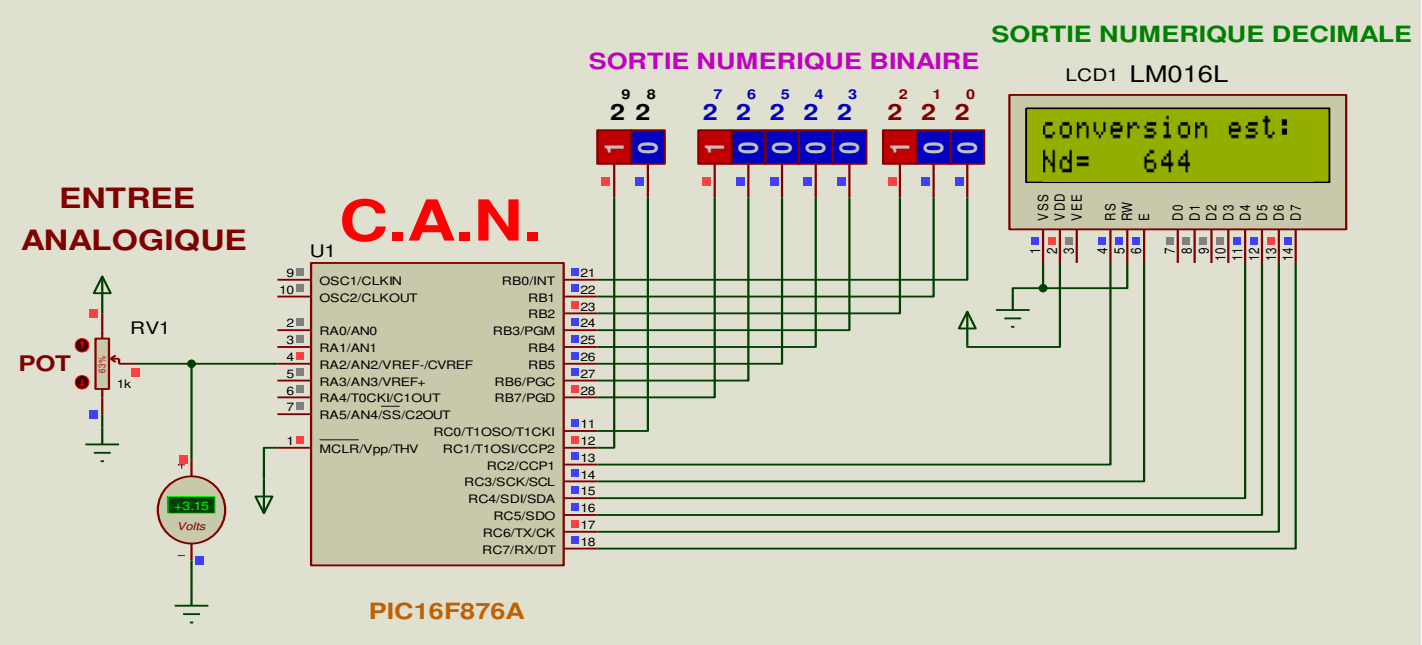
	Afficheur commandé : oui ou non				
Nombre à afficher	T1	T2	Afficheur 1	Afficheur 2	
	bloqué	bloqué	non	non	1ms
Unité de TMR0	saturé	…	oui	non	10ms
	bloqué	bloqué	non	non	1ms
Dizaine de TMR0	…	…	…	…	10ms
	bloqué	bloqué	non	non	1ms

3°) Compléter le programme :

```
program Exercice_22_TIMER0_comp_100;

program TIMER0;
 Var
 uni:byte;
 dix:byte;

begin
trisb:=$.....;            // de RB0 à RB3  sorties ,de RB4 à RB7 entrées
trisa:=$.....;            //  de RA0 et RA1 sorties ,RA2 à  RA4 entrées

TMR0:=……..;       // initialisation du timer 0 à la valeur 0
OPTION_REG := %.....   .;
while true do
begin

      While TMR0 < ……. do

       begin
     uni  := TMR0 …….. 10;              // Identifier le chiffre de l'unité du TIMER0
     dix  := TMR0 ……..10 ;              // Identifier le chiffre de dizaine de la variable i

          porta:=0;
          delay_ms(1);
          portb:=uni;
          porta:=……;                   //  affichage multiplexé puisqu'on dispose d'un seul décodeur
          delay_ms(10);
          porta:=0 ;
          delay_ms(1);
          portb:=dix;
          porta:=……..;
          delay_ms(10);

      end;
 TMR0:=…….
 end;
end.
```

Exercice N°23

Soit à convertir une tension comprise entre 0 et 5V fournit par un potentiomètre branché sur RA2, et afficher le résultat sous forme binaires avec des LEDs. (Justification à droite) ,et sous forme décimale sur un afficheur LCD .



Le PIC **16F876A** possède un convertisseur analogique numérique sur 10 bits,

Lorsque la tension varie de 0 à 5V , N varie de 0 à 1023

1°) Trouver la relation entre N et Ue

Ue ⟶ N

5V ⟶ 1023

N= ……

2°) Configurer les entrées /sorties :

TRISA

1	..	…	…	…	..

TRISB

..	..	..	..	..	..	..	..

TRISC.0= …. TRISC.1=…..

3°) Configurer le registre ADCON 1

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
…	0	0	0	…	…	…	…

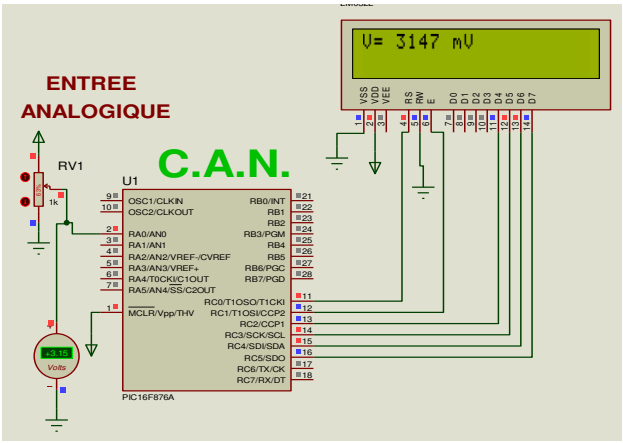
4°) Compléter le programme

```
program exercice_N_23_CNA;
var N : word; //  déclaration d'une variable N de type mot  sur 16bits
Nd:string[5]; //declaration d'une variable Nd de type chaine de caractéres
 // Connection module LCD
var LCD_RS : sbit at RC2_bit;
var LCD_EN : ……
var LCD_D4 : ……..
var LCD_D5 : …….
var LCD_D6 : ……
var LCD_D7 : …….

var LCD_RS_Direction : sbit at TRISC2_bit;
var LCD_EN_Direction : …….
var LCD_D4_Direction : ……
var LCD_D5_Direction : ……
var LCD_D6_Direction : sbit at TRISC6_bit;
var LCD_D7_Direction : sbit at TRISC7_bit;

 begin
   Lcd_init();  // initialiser le module LCD
   Lcd_Cmd(_LCD_CURSOR_OFF);
   ADCON1:=%.....;    // justification des 10 bits a droite et RA2 entrée ANA
   TRISA  := $.....;         // PORTA Entrées
   TRISB  := …….;           // PORTB Sorties
   TRISC.0  := ……..;           // la broche RC0 est cofigurée comme sortie
   TRISC.1  := …….;           //  la broche RC1 est cofigurée comme sortie
   Lcd_out(……,…..,'conversion est:'); // Ecrire « conversion est : » à la première ligne et premier colonne
   ……  // boucle infinie
     begin
       N := ADC_Read(……); // lecture de la valeur lue par le convertisseur sur le canal 2

       PORTB := ……..;  // Les 8 bits de plus faibles poids sont aux PORTB
       PORTC := N shr(…..);  // Afficher les 2 bits de fort poids sur  RC0 et RC1
       wordtostr(N,Nd);  // transformer la variable N de type word en chaine de caractéres
       Lcd_out(…..,….,'Nd='); // Ecrire « Nd= » à la deuxième  ligne et premier colonne
       Lcd_out(…..,……,Nd); // Ecrire Nd à la deuxième  ligne et cinquième colonne
         end;
 end.
```

Exercice N°24

Soit à convertir une tension variable de 0 à 5V branchée sur l'entrée RA0

L'affichage de la tension en mV est réalisé par un afficheur LCD comme le montre la figure ci-contre



1°) Comment doit-on configurer l'entrée RA0 ?
Numérique ou analogique

……

2°) Configurer le registre TRISA «Tout le PORTA est utilisé comme entrée»

TRISA

…	..	…	…	…	..

Le convertisseur C.A.N fournit un nombre binaire naturel de 10 bits (B9 B8 B7 B6 B5 B4 B3 B2 B1 B0)
Deux registres (2 X 8 bits) sont nécessaire pour stocker le résultat de la conversion. Ce sont les registres :

- ADRESH
- ADRESL

3°) Sachant que le résultat de la conversion est justifié à droite compléter les deux registres ADRESH et ADRESL

ADRESH

0	…	…	…	…	…	…	…

ADRESL

…	…	…	…	…	…	…	…

10 bits du résultat

3°) Configurer le registre ADCON 1

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
…	**0**	**0**	**0**	**…**	**…**	**…**	**…**

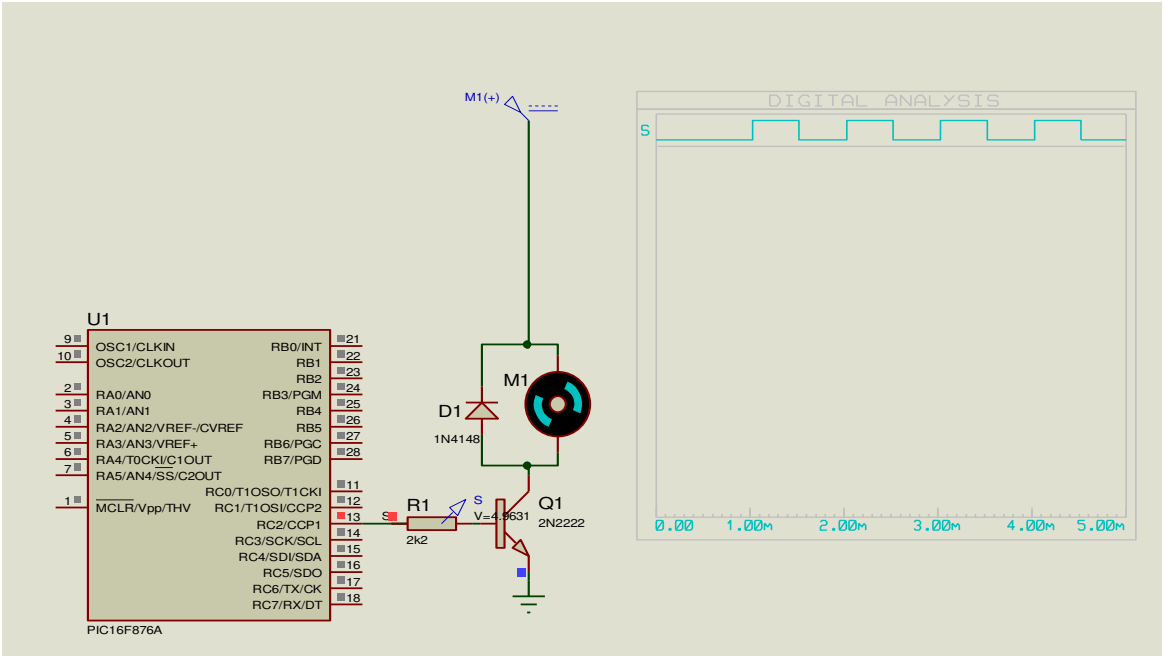
4°) Compléter le programme ci-dessous :

```
program exercice_N_24_voltmétre;

var

 N : word ;    // 2  octets car le résultat de conversion est sur 10 bits

 calcul:real ;  // 4  pour ne pas avoir un dépassement de taille lors de la multiplication

 tension:word ;   // 2  octets car la tension est affichée en mV elle est compris entre 0 et 5000 mV

 valeur_affichage :string[4]; // chaine de 4 caractéres pour afficher la tension

 // connection de L'LCD
```

```
    LCD_RS :sbit at portc.0;
    LCD_EN :……
    LCD_D4 : ……
    LCD_D5 : ……
    LCD_D6 : ……
    LCD_D7 : ……
    LCD_RS_direction :sbit at TRISC.0;
    LCD_EN_direction : ……
    LCD_D4_direction : ……
    LCD_D5_direction : ……
    LCD_D6_direction : ……
    LCD_D7_direction : ……
begin
   TRISA:=$.....;
   ADCON1:=%    .....        ;
   LCD_init();
   LCD_cmd(_LCD_cursor_off);
   LCD_out (1,1,'V=');
    while true do
    begin
         N:= ADC_read(…….); // ……
         calcul:= (N*(5000/1023));
         tension := word(calcul);  // ……..
         wordtostr(tension,valeur_affichage); // …….
         LCD_out (1,3,valeur_affichage);
…….. // Afficher au premier ligne et 9ème colonne 'mV'
         delay_ms(1000);
         end;
end.
```

Exercice N°25 :

Ecrire un programme qui permet de commander un moteur à courant continu avec un rapport cyclique **α = 0,5**. Utiliser la technique MLI (PWM)
Sortie sur **RC2** fréquence de MLI (**1000Hz)**

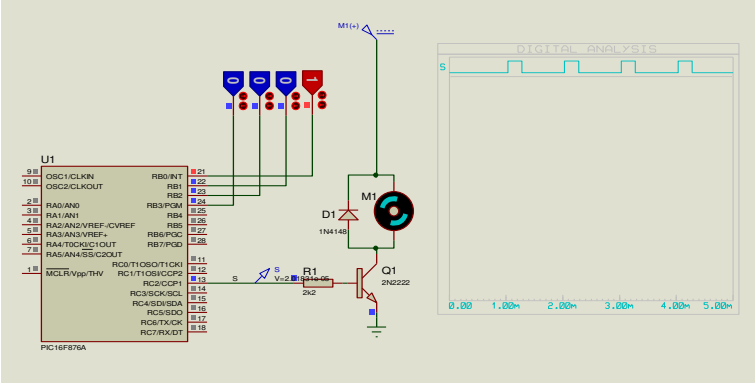



```
program exercice_N_25_MLI;
begin
PWM1_init(500); // initialisation du  PWM à 1000Hz
PWM1_start;
while true do
begin
PWM1_Set_Duty(……);
end;
end.
```

Exercice N°26 :

Ecrire un programme qui permet de commander un moteur à courant continu avec **4 vitesses**
Sortie sur **RC2** fréquence de MLI (**1000Hz)**
Compléter le tableau ci-dessous et le programme

Entrées	Rapport cyclique	N
PORTB =0	α = 0	N = 0
PORTB =1	α = 0,25	N = 64
PORTB =3	α =0,5	N =….
PORTB =7	α =0,75	N =….
PORTB =15	α =1	N =…



```
program exercice_N_26_MLI;

begin

  TRISB:=$FF;

  PWM1_Init(……..);

  PWM1_Start;

  while true do

  begin

  if PORTB=0 then PWM1_Set_duty(……);

  if PORTB=1 then PWM1_Set_duty(……);

  if PORTB= 3 then PWM1_Set_duty(……);

  if PORTB= 7 then  PWM1_Set_duty(……);

  if PORTB= 15 then PWM1_Set_duty(255);

  end;

end.
```

Exercice N°27 :

Ecrire un programme qui permet de commander un moteur à courant continu avec un rapport cyclique variable de 0 à 1 en utilisant la technique MLI (PWM)

Le rapport cyclique augmente si on appui sur le bouton RA2 et il diminue si on appui sur le bouton RA1

Utilisation de la fonction **Button** (port,bit,temps d'appui en ms ,état logique actif)

Exemple : if Button (portA,3,50,1) then « action1 »

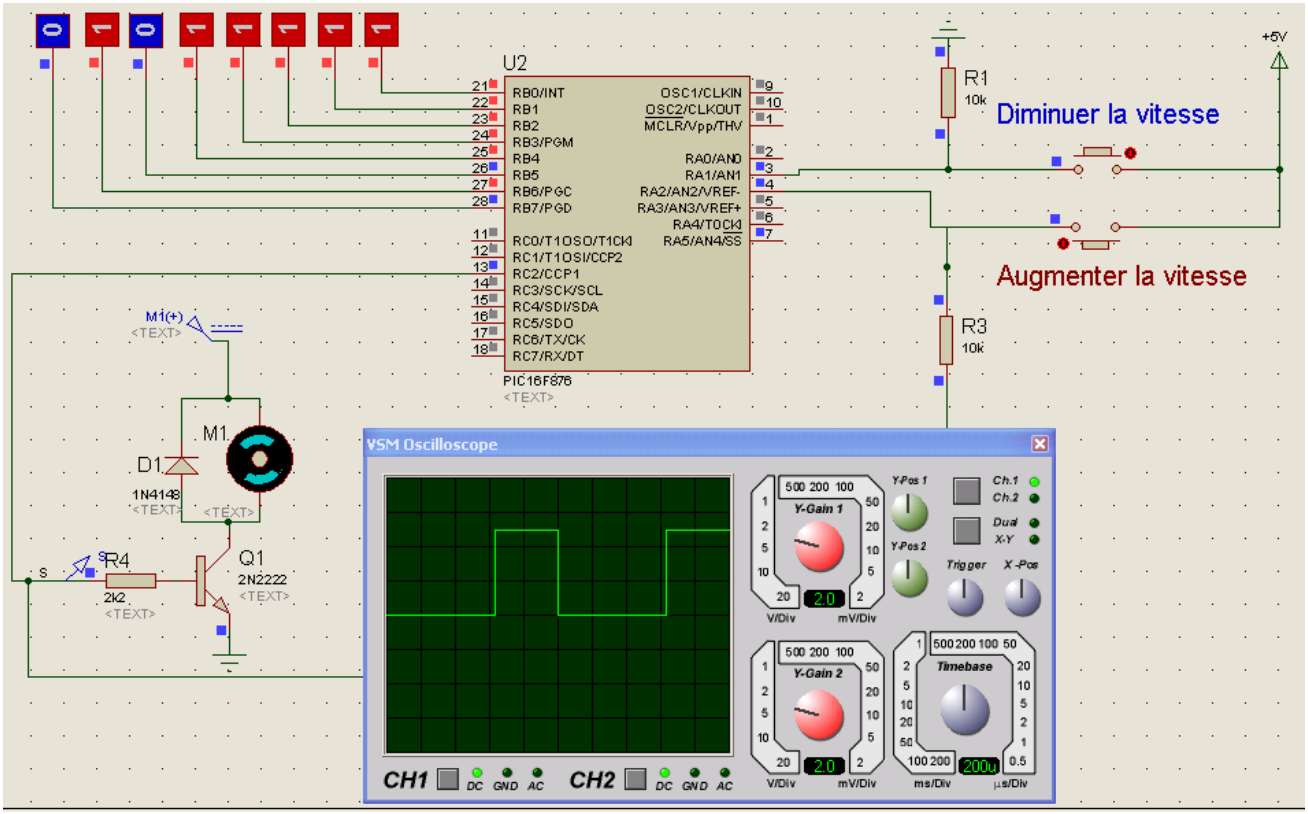
On teste l'appui sur un bouton poussoir relié à la broche RA3 pendant 50ms pour faire l'action 1

1°) Compléter les phrases suivantes :

Si on appui sur le bouton RA2 la vitesse du moteur ……

Si on appui sur le bouton RA1 la vitesse du moteur ……

2°) Compléter le programme :




```
program exercice_N_27_MLI;
var x:byte;
begin
 PWM1_Init(1000);
 PWM1_Start();
ADCON1:=$.....;  // PORTA numérique
 trisa:=$.....;trisb:=…….;portb:=…….;x:=0;
 while true do
 begin
  if button(porta,2,100,1) then INC(x); if x=255 then dec(x);
  if button(porta,1,100,1) then DEC(x); if x=0 then inc(x);
  portb:= x;
  PWM1_Set_duty(…..);
  end;
  end.
```

3°) Expliquer le rôle des deux instructions colorées en bleu.

Exercice N°28 :

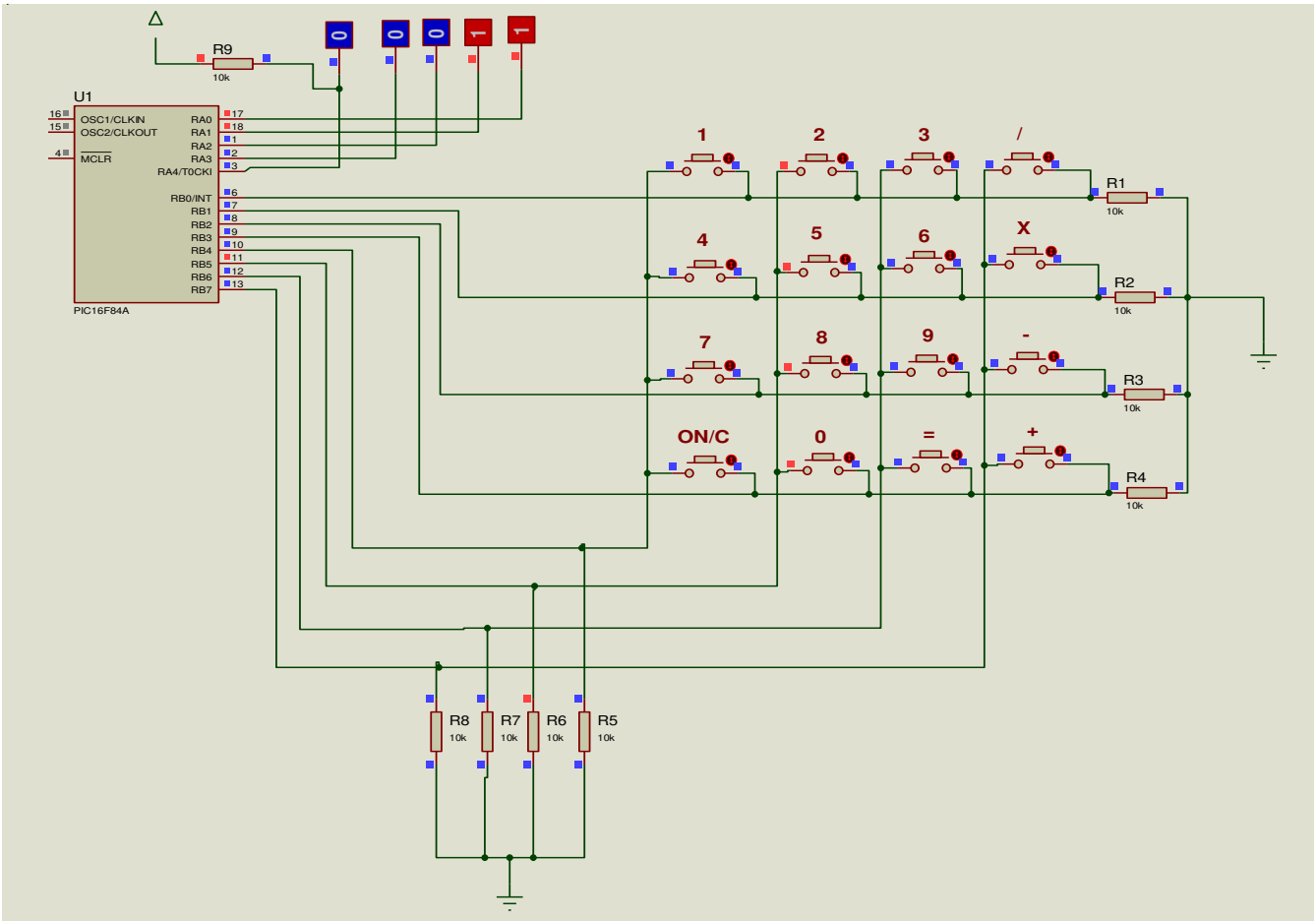
Lire le programme ci-dessous puis remplir les tableaux 1 , 2 et 3 :

```
program exercice_N_28_Clavier;
var keypadPort : byte at PORTB; // le clavier est relié au port B
var kp : byte;  // On déclare une variable kp de type octet
begin
trisa:=0;  // Le portA est configuré en sortie
porta:=0;  // Initialisation du portA
Keypad_Init(); // initialiser le portB pour communiquer avec le clavier
while true do
  begin
   kp := Keypad_Key_Click();
   if kp <> 0 then porta:=kp;
  end;
end.
```

Tableau 1

Touche	Kp	PortA
1	…	…
2	…	…
3	…	…
4	…	…
5	…	…
6	…	…
7	…	…
8	…	…
9	…	…
0	…	…
ON/C	…	…
+	…	…
/	…	…
=	…	…
-	…	…
x	…	…

Montage 1



Montage 2

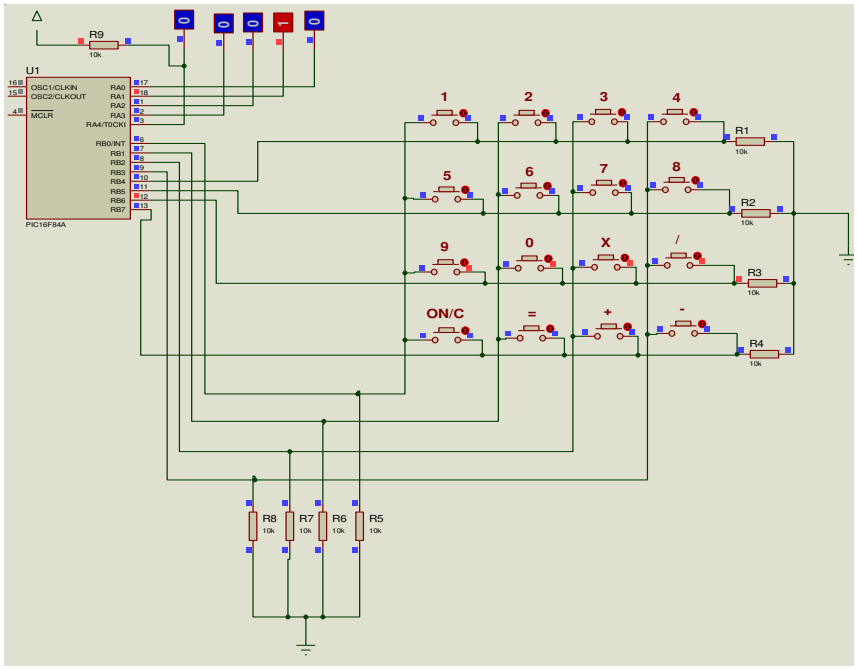


Tableau 2

Touche	Kp	PortA
1	…	…
2	…	…
3	…	…
4	…	…
5	…	…
6	…	…
7	…	…
8	…	…
9	…	…
0	…	…
ON/C	…	…
+	…	…
/	…	…
=	…	…
-	…	…
x	…	…

Montage 3

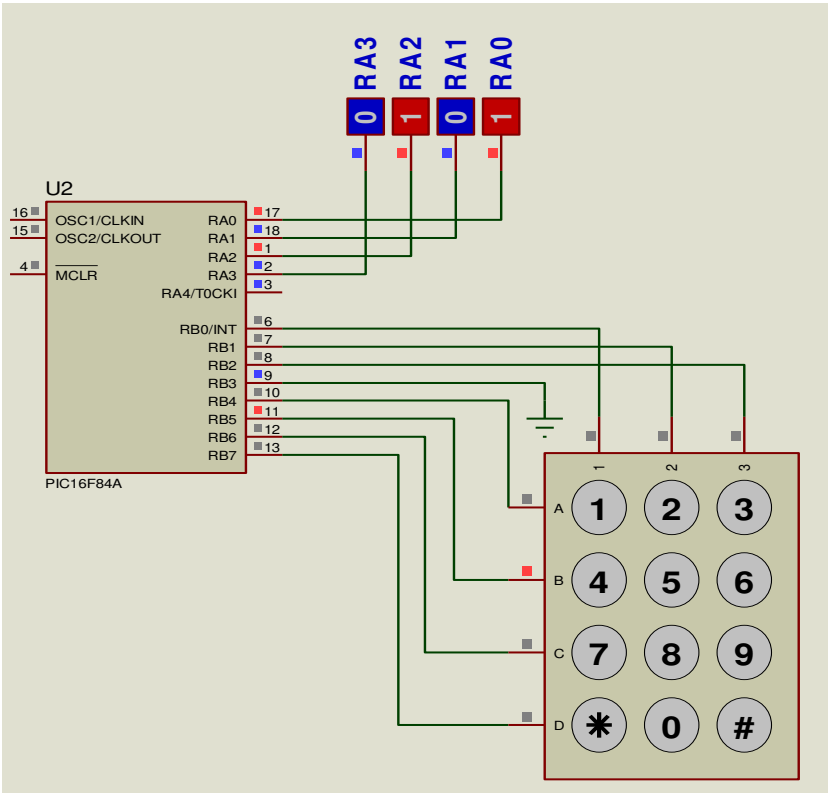


Tableau 3

Touche	Kp	PortA
1	…	…
2	…	…
3	…	…
4	…	…
5	…	…
6	…	…
7	…	…
8	…	…
9	…	…
*	…	…
0	…	…
#	…	…

REGITRE DE CONFIGURATION DES INTERRUPTIONS (INTCON) :

Le registre INTCON (INTerrupt CONtroller) est le registre principal de contrôle et de gestion des interruptions.

Le registre INTCON est parfois différent d'un PIC à un autre il est impératif de revenir au document constructeur pour chaque type de microcontrôleur.

Registre **INTCON** **pour PIC16F84A**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	**EEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**

Registre INTCON

GIE : « *Global Interrupt Enable* » mis à 1 autorise toutes les interruptions non masquées par leur bit individuel.

EEIE : « *EEPROM write completed Interrupt Enable » :* autorise les interruptions de fin d'écriture dans l'EEPROM.

T0IE : « *Timer 0 Interrupt Enable » :* mis à 1 autorise les interruptions dues au débordement du *timer* 0.

INTE : « *INTerrupt Enablé » :* mis à 1, autorise les interruptions sur RB0/INI. L'interruption a lieu sur le front montant de l'impulsion si le bit INTEG (*INTerrupt Edge*) du registre OPTION est à 1 ; elle a lieu sur le front descendant si ce bit est à 0.

RBIE : « RB *Interrupt Enable* » :mis à 1, autorise les interruptions sur RB4 à RB7.

T0IF : « *Timer* 0 *Interrupt Flag » :* est mis à 1 en cas de débordement du *timer* 0.

INTF : « *INTerrupt Flag » :* est mis à 1 si une interruption est générée sur RB0/INT.

RBIF : « RB *Interrupt Flag »,* est mis à 1 lors d'un changement d'état sur une des lignes RB4 à RB7.

Chaque indicateur de changement d'état doit être remis à 0 par le logiciel dans le programme de traitement de l'interruption.

NB : Lors d'un *Reset,* tous les bits du registre INTCON sauf RBIF sont mis à 0. RBIF garde son état précédent.

LE TIMER TMR0

Le registre TMR0 est un compteur programmable de 8 bits (de 0 à 255).

La configuration du TMR0 est assurée par le registre OPTION « **OPTION_REG** »

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	**INTEDG**	**TOCS**	**TOSE**	**PSA**	**PS2**	**PS1**	**PSO**

OPTION_REG

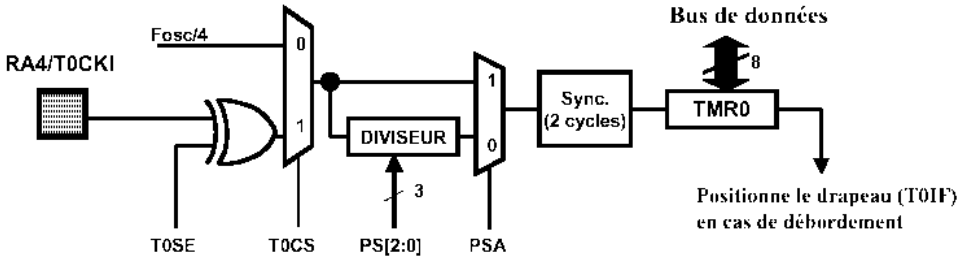


Schéma synoptique du registre OPTION

PS2	PS1	PSO	Diviseur
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Tableau 1

Le TMR0 est incrémenté en permanence soit par :

- L'horloge interne (**fosc/4**) « **mode TIMER** »
- L'horloge externe appliquée à la broche **RA4** du portA « **mode compteur** »

Le choix de l'horloge se fait à laide du **bit 5** du registre **OPTION_REG « TOCS »**

- **TOCS = 0** Horloge interne « **mode TIMER** »
- **TOCS = 1** Horloge externe « **mode COMPTEUR**»

Dans le cas de l'horloge externe ,**le bit 4 « TOSE »** du registre **OPTION_REG** permet de choisir le **front** sur lequel le **TIMER0** s'incrémente :

- **TOSE = 0** incrémentation **sur fronts montants**
- **TOSE = 1** incrémentation **sur fronts descendants**

Quelque soit l'horloge choisie, on peut la faire passer dans un diviseur de fréquence programmable (prescaler) dont le rapport est fixé par les bits **PS0,PS1 et PS2** du registre **OPTION_REG** « voir tableau 1 »

L'affectation ou non du prédiviseur se fait à l'aide du **bit 3 « PSA »** du registre **OPTION_REG**

- **PSA =0** on utilise le prédiviseur.
- **PSA =1** pas de prédiviseur.

Bit 6 :INTEDG « INTerrupt Edge » : dans le cas où on utilise l' interruption externe avec RB0

- Si **INTEDG** = 1, on a interruption si le niveau sur RB0 passe de 0 vers 1. « front montant »
- Si **INTEDG** = 0, l'interruption s'effectuera lors de la transition de 1 vers 0. « front descendant »

Bit 7 : $\overline{\text{RBPU}}$: Quand ce bit est mis à 0, une résistance de rappel au +5 volt est placée sur chaque broche du PORTB

N.B À l'issue d'un *Reset,* le registre OPTION_REG = 11111111

REGISTRE ADCON1

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
bit 7	bit 6			bit 3	bit 2	bit 1	bit 0

Bit 6, bit 5 et bit 4 **:** Bits non implantés.

Bit 3,bit 2,bit 1 et bit 0 **:**PCFG3, PCFG2, PCFG1 et PCFG0 : bits de contrôle de la configuration des ports :

Ces bits permettent de choisir le partage entre entrées analogiques et digitales sur les ports A et E .

Ils permettent également de choisir pour Vref+ entre Vdd et RA3 et pour Vref- entre Vss et RA2 selon le tableau suivant :

4 bits PCFG				PIC 16F877 / PIC 16F876								Tensions De références	
				PORTE			PORTA						
PCFG3	PCFG2	PCFG1	PCFG0	AN7/RE2	AN6/RE1	AN5/RE0	AN4/RA5	AN3/RA3	AN2/RA2	AN1/RA1	AN0/RA0	Vref+	Vref-
0	0	0	0	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0	0	0	1	A	A	A	A	Vref+	A	A	A	RA3	V_{SS}
0	0	1	0	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}
0	0	1	1	D	D	D	A	Vref+	A	A	A	RA3	V_{SS}
0	1	0	0	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}
0	1	0	1	D	D	D	D	Vref+	D	A	A	RA3	V_{SS}
0	1	1	X	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}
1	0	0	0	A	A	A	A	Vref+	Vref-	A	A	RA3	RA2
1	0	0	1	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}
1	0	1	0	D	D	A	A	Vref+	A	A	A	RA3	V_{SS}
1	0	1	1	D	D	A	A	Vref+	Vref-	A	A	RA3	RA2
1	1	0	0	D	D	D	A	Vref+	Vref-	A	A	RA3	RA2
1	1	0	1	D	D	D	D	Vref+	Vref-	A	A	RA3	RA2
1	1	1	0	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}
1	1	1	1	D	D	D	D	Vref+	Vref-	D	A	RA3	RA2

A : entrée analogique **D** : entrée numérique $V_{DD} = V_{CC} = 5V$; V_{SS} = GND = 0 V

Au reset ADCON1 = 00000000 : cela signifie que les 5 bits de port A et les 3 bits de Port E sont configurés en entrées analogiques.

Pour récupérer les 5 bits du port A et les trois bits du port E en tant que I/O numériques (digitales) il faut écrire la valeur **'0 6'** dans ADCON1

N.B : On s'intéressera uniquement au cas où Vref- = V_{SS} = 0 et Vref+=V_{DD} = 5V cas des lignes coloriées en rose dans le tableau.

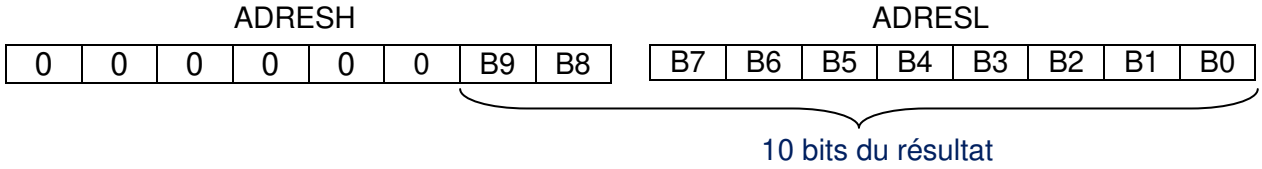
Bit 7 : ADFM

Le convertisseur C.A.N fournit un nombre binaire naturel de 10 bits (B9 B8 B7 B6 B5 B4 B3 B2 B1 B0)
Deux registres (2 X 8 bits) sont nécessaire pour stocker le résultat de la conversion. Ce sont les registres :

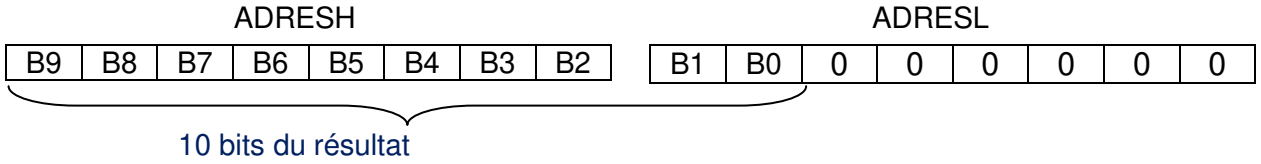
- ADRESH
- ADRESL

Deux formats sont disponibles suivant la valeur du bit ADFM :

ADFM =1 :le résultat de la conversion est justifié à droite :ADRESH ne contient que les 2 MSB du résultat . Les 6 MSB de ce registre sont lus comme des « 0 »



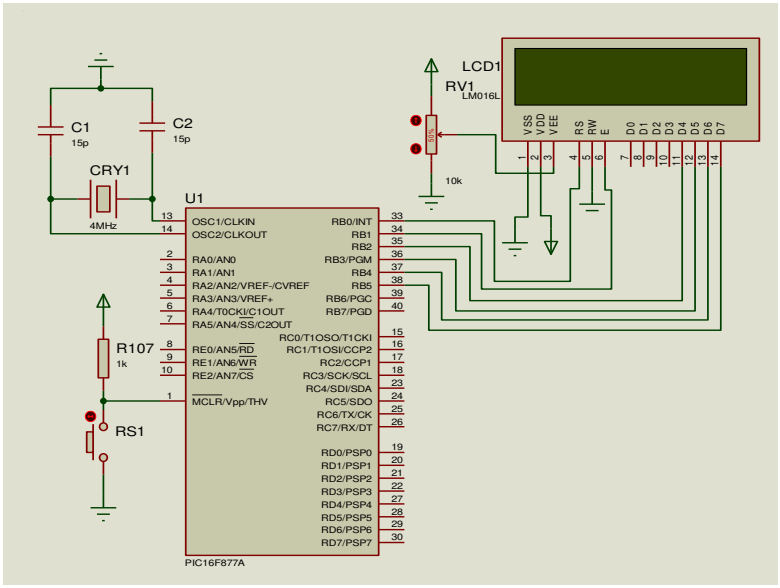
ADFM =0 :le résultat de la conversion est justifié à gauche :ADRESL ne contient que les 2 LSB du résultat . Les 6 LSB de ce registre sont lus comme des « 0 »



INSTRUCTIONS SPECIFIQUE AU COMPILATEUR MIKROPASCAL PRO POUR L'AFFICHEUR LCD

Les variables suivantes doivent être définies dans tous les projets utilisant la bibliothèque d'affichage à cristaux liquides LCD:

```
// Connections du module LCD
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;
 // FIN
```



Lcd_Init () ; // Initialisation de l'LCD

Lcd_Out(1, 2, 'BRAVO') ; // écrire BRAVO sur l'LCD à partir de la ligne 1 ,colonne 2

Lcd_Chr(2, 3, i); // écrire la caractère équivalent en code ASCII i sur l'LCD à partir de la ligne 2 ,colonne 3

Lcd_Cmd exemples :

Lcd_Cmd(_LCD_CLEAR); // effacer l'LCD

Lcd_Cmd(_LCD_CURSOR_OFF) ; // supprimer le curseur

INSTRUCTIONS SPECIFIQUE AU COMPILATEUR « MIKROPASCAL PRO » POUR LE CLAVIER

Var keypadPort : byte at PORT… ; // Pour la connexion du clavier au PORT. .. considéré (8 bits)

Var Kp :byte ; // on définit une variable de type octet

Keypad _Init(); // Initialisation du clavier

Kp:=Keypad_key_Press(); // lecture de code de la touche « touche enfoncée » de 1 à 16.

Kp:=Keypad_key_click(); // lecture de code de la touche « touche enfoncée

INSTRUCTIONS SPECIFIQUE AU COMPILATEUR MIKROPASCAL PRO POUR LE MODULE CONVERSION

ADC_Init() ; // Initialise le module convertisseur et le configurer avec les réglages suivants:

Vref-=0 ;Vref+ = 5V , Utilisation de l'horloge interne pour la conversion.

N : word // déclaration d'une variable de type word

N := ADC_Get_Sample(1) // lecture de la valeur lue par le convertisseur sur le canal 1

N := ADC_Read (2) // lecture après initialisation et démarrage de la conversion sur le canal 2

INSTRUCTIONS SPECIFIQUE AU COMPILATEUR MIKROPASCAL PRO POUR LE MODULE PWM

PWMx_Init(1000) // Initialise le module PWM de la sortie CCPx à la fréquence 1000Hz:

PWMx_start() // Démarrage du module PWM et sortie du signal sur la broche CCPx

PWMx_Set_duty(N) // Change le rapport cyclique α du signal sortant sur la broche CCPx avec

N variant de 0 à 255 $\alpha = \frac{N}{255}$

PWMx_stop // Arrêter le module PWM de la sortie CCPx