Exercice N°1

1°)

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

2°)

TRIS A				1	1	1	1	1	= (1F)Hex
TRIS B	1	1	1	0	0	0	0	0	= (E0)Hex

3°) Les équations des sorties :

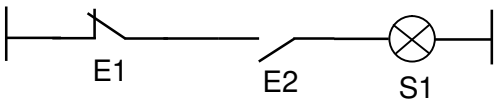
$S1 = \overline{E1} \bullet E2 \bullet \overline{E3} \bullet E4$ $S2 = E1 \oplus E4$ $S3 = E1 \bullet \overline{E2} + E3 \bullet \overline{E4}$ $S4 = E2 \odot E3$ $S5 = \overline{E1 + E3}$

4°) Programme MikroPascal correspondant aux équations précédentes :

```
program exercice1_equation;
  var
E1:Sbit at RA0_bit;
E2:Sbit at RA1_bit;
E3:Sbit at RA2_bit;
E4:Sbit at RA3_bit;
S1:Sbit at RB0_bit;
S2:Sbit at RB1_bit;
S3:Sbit at RB2_bit;
S4:Sbit at RB3_bit;
S5:Sbit at RB4_bit;
Begin
Trisa:=$1F ; // tout le portA est configuré comme entrées
Trisb:=$E0 ; // RB0,RB1,RB2,RB3,RB4,sorties  RB5,RB6,RB7 entrées
Portb :=0 ;       // initialisation
While true do
       Begin
S1:=not E1 and E2 and not E3 and E4 ;      // equation de S1
S2:= E1 xor E4   ;    // equation de S2
S3:= E1 and not E2 or E3 and not E4 ;    // equation de S3
S4:=  not (E2 xor E3);    // equation de S4
S5:= not(E1 or E3);    // equation de S5
End ;
END .
```

5°) Si on modifier le microcontrôleur PIC 16F84A par un PIC 16F628A l'instruction à ajouter au programme précédent pour programmer ces équations est : **CMCON :=$07 ;**
Pour désactiver les comparateurs analogiques et pour rendre le port A numérique

Exercice N°2:



1°) Déduire l'équation logique de RB0

$$S1 = \overline{E1} \bullet E2$$

2°) Programme:

```
program exercice2_equation;
Var
E1: sbit at porta.0;
E2: sbit at porta.1;
S1: sbit at portb.0;
  Begin
Trisa:=$1F ;
Trisb:=$00 ;
 CMCON:=$07;  // désactivation des comparateurs analogique et PORTA numérique
Portb :=0 ;        // initialisation
While true do   // boucle infinie
    Begin
  S1:= not E1 and E2    ;
     End;
 End.
```

Exercice N°3:

1°) Equation de X $X = \overline{S1} \bullet (S2 + X)$

2°) Equation de Y $Y = \overline{S4} \bullet Y + S3$

3°) Programme

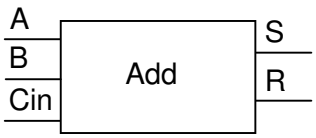
```
    program exercice_3_fonctions_memoires;
var
S1: sbit at RA0_bit;
S2: sbit at RA1_bit;
S3: sbit at RA2_bit;
S4: sbit at RA3_bit;
 X,Y:bit;
      Begin
            X:=0;Y:=0;
           Trisa:=$ 1F ;
           Trisb:=$ 00  ;
```

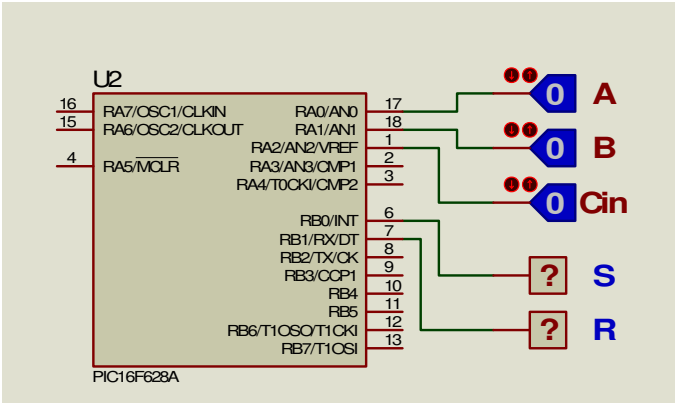
```
        Portb := 0 ;  // initialisation
   while true do
       Begin
        if ((S1=0) and ((S2=1) or  (X=1))) then X:=1  else X:=0;
          if ((S4=0) and (Y=1)  or  (S3=1)) then Y:=1  else Y:=0 ;
          if X=0 then portb.0:=0 else portb.0:=1;
         if Y=0 then  portb.1:=0 else portb.1:=1  ;
end;
end.
```

Exercice N° 4: (Additionneur complet)

On désire réaliser un additionneur complet avec le circuit 16F628A

Le circuit possède 3entrées et deux sorties :





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

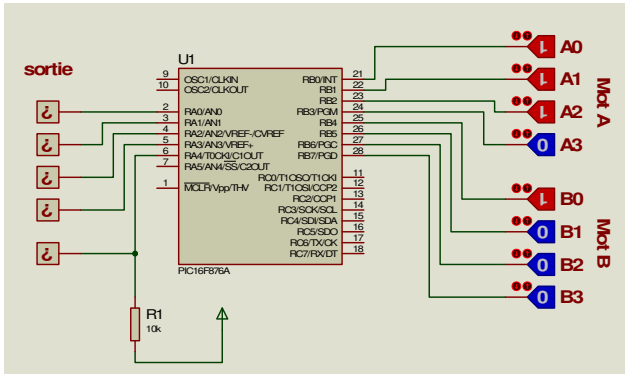
$R1 = .(A \oplus B).Cin + A.B$

Programme:

```
program Additonneur;
  var
A:   sbit at PORTA.0;  // variable de type bit affecté au PORTA.0 çad "RA0"
B:   sbit at PORTA.1;
Cin: sbit at PORTA.2;
S:   sbit at PORTB.0;
R:   sbit at PORTB.1;
begin
  TRISA:=$FF;
  TRISB:=$00 ;
CMCON:=$07;  // Désactivation des comparateurs " PORTA numérique "
  PORTB:=0; // initialisation des sorties
  while true do begin
  S:=  (A xor B) xor Cin ;  // équation de la somme
  R:= (A xor B) and Cin or A and B;  // equation du retenu
    end;
end.
```

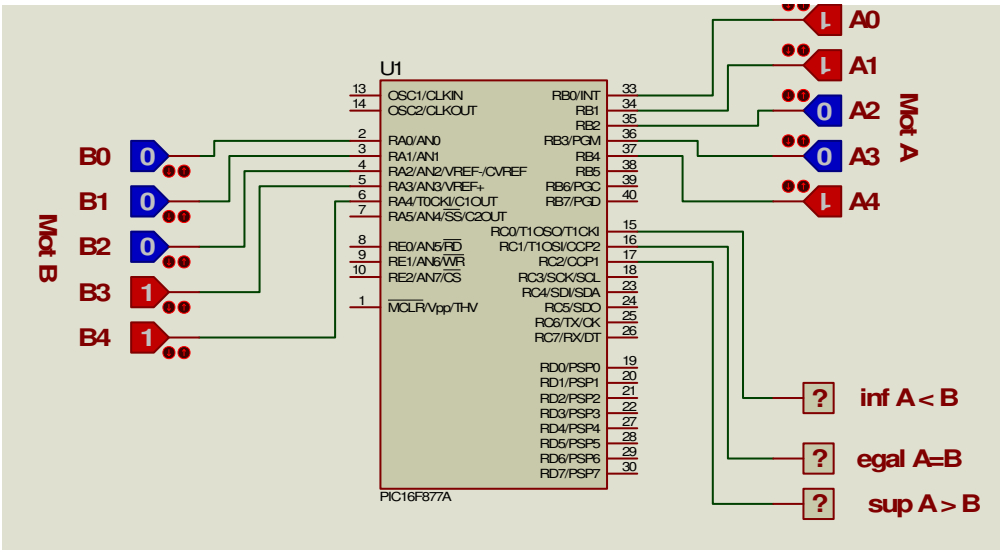
Exercice N° 5: (Additionneur BCD)

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:=$00;  //
  TRISB:=$FF;  //
   PORTA:=0;  //
   while 1=1 do  //
   begin
   A:=PORTB;  //  La variable A  reçoit  le contenu du PORTB
   A:= A and $0F;  //  Masquer les 4 bits de poids le plus fort
   B:=PORTB;  //  La variable A  reçoit  le contenu du PORTB
   B:=B shr 4 ;  //  Décalage à droite de 4 bits
   S1:= A+ B ;  //  Addition de A et B
   if S1 > 9 then S1:= S1 + 6 ;  // Correction  si  la somme est > 9
   PORTA:= S1;
   end;
end.
```

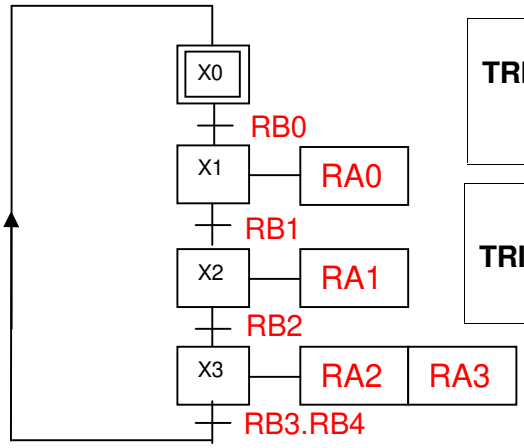
Exercice N° 6: (comparateur)




```
program comparateur;
var
A:byte at portB;
B:byte at portA;
inf:sbit at portc.0;
ega:sbit at portc.1;
sup:sbit at portc.2;
begin
  TRISA:=$FF;
  TRISB:=$FF;
  TRISC:=$00;
  PORTC:=0;
  ADCON1:=$07;
  while true do
  begin
   if A < B then inf:=1 else inf:=0;
   if A > B then sup:=1 else sup:=0;
   if A = B then ega:=1 else ega:=0;
    end;
end.
```

Exercice N°7:

1°) GRAFCET codé microcontrôleur



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

TRIS A				RA4	RA3	RA2	RA1	RA0
				0	0	0	0	0

TRIS B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	1	1	1	1	1	1	1	1

3°) Programme

```
program exercice7_grafcet1;
      Var
  m: sbit at RB0_bit ;
  S1: sbit at RB1_bit ;
  S2: sbit at RB2_bit ;
  S3: sbit at RB3_bit ;          //  Déclaration des variables
  S4: sbit at RB4_bit ;
  KM1: sbit at RA0_bit ;
  KM2: sbit at RA1_bit ;
  KM3: sbit at RA2_bit ;
  KM4: sbit at RA3_bit ;
  X0,X1,X2,X3:bit;

                          BEGIN
          TRISA :=$00 ;          // Configuration du registre TRISA en Hexadécimal
          TRISB :=$FF;           // Configuration du registre TRISB en Hexadécimal
          PORTA:=0 ;             // Initialisation du portA
 X0:=1 ; X1:=0 ; X2:=0 ; X3:=0 ;  // Initialisation des variables
              While true do      // boucle infinie
          BEGIN
  IF ((X0=1) AND (m=1)) THEN          // Condition d'activation de l'étape1
                          BEGIN
                            X0:=0;X1:=1 ;
                           END ;
  IF ((X1=1) AND (S1=1)) THEN         // Condition d'activation de l'étape2
                          BEGIN
                          X1 := 0 ; X2 := 1 ;
                         END ;
   IF ((X2=1) AND (S2=1)) THEN        // Condition d'activation de l'étape3
```

```
                BEGIN
                X2:=0;X3:=1
                END ;
    IF ((X3=1) AND (S3=1) AND  (S4=1) ) THEN        // Condition d'activation de l'étape0
                BEGIN
                X3:=0;X0:=1 ;
                END ;
IF (X1=1) THEN KM1:= 1 ELSE KM1:= 0 ;                  // Programmation de la sortie KM1
IF (X2=1) THEN KM2:= 1 ELSE KM2 := 0 ;                  // Programmation de la sortie KM2
IF (X3=1) THEN KM3 := 1 ELSE KM3:= 0 ;                 // Programmation de la sortie KM3
IF (X3=1) THEN KM4 := 1 ELSE KM4 := 0 ;                //Programmation de la sortie KM4
        END ;
   END.
```

ExerciceN°8

1°) Affectation des deux registres **TRISA** et **TRISB**.

TRISA				RA4	RA3	RA2	RA1	RA0
				1	1	1	1	1

TRISB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	0	0	0	0	0	0	0	0

2°) Programme en langage pascal relatif à la commande du système :

```
program Exercice8_grafcet2;
Var
dcy:sbit at porta.0;
S1:sbit at porta.1;
S2:sbit at porta.2;
S3:sbit at porta.3;
MT:sbit at portb.0;
Se:sbit at portb.1;
 var  X0, X1, X2, X3, X4,X5,X6,T1,T2:bit;
   begin
   trisa := $1F; //  configuration  du portA en entrée
   trisb := $00; portB := 0; // configuration et initialisation du portB
   X0:=1; X1:=0 ;X2:=0;X3:=0; X4:=0;X5:=0;X6:=0;
       while true do       // boucle infinie
       begin
       if ((X0=1) and (dcy=1)) then
                begin
                     X0 := 0; X1 := 1;
                end;
     if ((X1=1)  and ( S2=1) and ( S1 =0)) then
               begin
                  X1 := 0; X2 :=1;
               end;
     if((X2=1)  and ( S3=1) ) then
```

```
            begin
            X2 := 0; X3 :=1;
           end;
      if ((X3=1) and ( S3 =0)) then
           begin
     X3 := 0; X4 :=1;
         end;

     if ((X1=1)  and (S1=1) and( S2=1)) then
            begin
              X1 := 0; X5 :=1;
            end;
   if((X5=1)  and (S3=1))   then
           begin
             X5 := 0; X6 :=1;
           end;
     if  ((X6=1) and (t2=1) or (X4=1) and (t1=1))  then
                 begin
           X6 := 0; X4 :=0;X0:=1
             end;
   // programmation des sorties
    if((X1=1) or (X2=1) or (X3=1) or (X4=1) or (X5=1) or (X6=1))  then MT:=1 else MT:=0;
    if ((X3=1) or (X4=1)) then Se:=1 else Se:=0;
    // programmation des temporisations
if (X4=1)  then  T1 := 1 else T1:=0;
if T1=1 then delay_ms(5000);
if (X6=1)  then  T2 := 1 else T2:=0;
if T2=1 then delay_ms(6000);
       end;
  end.
```

ExerciceN°9

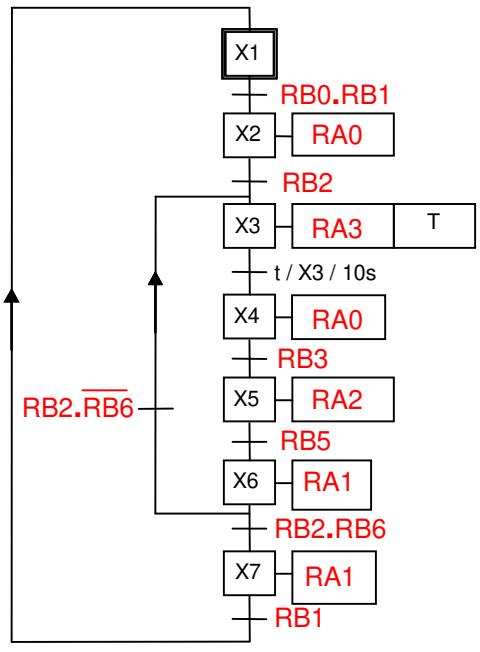
Système : déplacement d'un chariot:

1°) GRAFCET codé microcontrôleur

2°) Affectations des deux registres TRISA et TRISB.

	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
TRISA	0	0	0	0	0	0	0	0

	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
TRISB	1	1	1	1	1	1	1	1



3°) Programme:

```
program EXERCICE_9GRAFCET3;
Var
dcy:sbit at portB.0;
S1:sbit at portB.1;
S2:sbit at portB.2;
S3:sbit at portB.3;
L11:sbit at portB.5;
n:sbit at portB.6;
KM1:sbit at portA.0;
KM2:sbit at portA.1;
v14M1:sbit at portA.2;
KA:sbit at portA.3;
 var  X1, X2, X3, X4,X5,X6,X7, T : bit;
    begin
CMCON:=$07; // désactiver les comparateurs analogiques
                    et rendre le porta numérique
  trisA := $00; trisB := $FF; portA := 0;
X1 :=1; X2:=0;X3:=0;X4:=0;X5:=0;X6:=0;X7:=0;
         while (1=1) do
         begin
 if ((X1=1) and (Dcy=1) and( S1 =1)) then
                  begin
                       X1 := 0; X2 := 1;
                  end;
 if ((X2=1)  and ( S2=1) or (X6=1)  and (S2=1)  and(n=0))  then
                  begin
                    X2 := 0; X6 := 0; X3 :=1;
                  end;
if ((X3=1)and (t=1))   then
                 begin
                 X3 := 0; X4 :=1;
                end;
 if ((X4=1) and ( S3 =1)) then
                 begin
          X4 := 0; X5 :=1;
               end;
  if ((X5=1) and (I11=1)) then
                 begin
                  X5 := 0; X6 :=1;
                end;
  if ((X6=1)  and ( S2=1) and( n=1  )) then
                  begin
                    X6 := 0; X7 :=1;
                  end;
  if ((X7=1)  and ( S1=1)) then
                 begin
                    X7 := 0; X1 :=1;
                end;
```

```
// programmation des sorties

// programmation de la sortie KM1

if(X2=1)or (X4=1) then KM1:=1 else KM1:=0;

// programmation de la sortie KM2

if(X6=1)or (X7=1) then KM2:=1 else KM2:=0;

// programmation de la sortie 14M1

if(X5=1) then v14M1:=1 else v14M1:=0;

// programmation de la sortie KA

if(X3=1) then KA:=1 else KA:=0;



// programmation du temporisation

 if (X3=0)  then  t := 0 else

   begin

   delay_ms(10000); t := 1;
     end;
         end;
     end.
```

ExerciceN°10

1°) Tableau résumant le fonctionnement des diodes

PORTA	RA2	RA1	RA0	Etat des diodes
0	0	0	0	Eteintes
1	0	0	1	Clignotent pendant 2s
2	0	1	0	Chenillard décalage à droite
3	0	1	1	Allumées
4	1	0	0	Chenillard décalage à gauche
5	1	0	1	Eteintes
6	1	1	0	Allumées
7	1	1	1	Clignotent pendant 2s

2°) Valeurs des TRISA et TRISB du microcontrôleur :

TRISA	1	1	1	1	1	1	1	1	= (11111111)$_{Bin}$	= (FF)$_{HEX}$	= (255)$_{10}$
TRISB	0	0	0	0	0	0	0	0	= (00000000)$_{Bin}$	= (00)$_{HEX}$	= (0)$_{10}$

```
program exercice_N_10;

    var i:byte;  // declaration d'une variable i de type octet
    begin
                TRISB:= 0  ;      // Configuration du port B comme sortie
               TRISA:=$FF;  // Configuration du port A comme entrée
                PORTB:=0;    //  initialisation du port B
               CMCON:=$07;  //  Désactivation du comparateur ,PORTA numérique
     While true do     //    Boucle infinie
          begin
           if  ((PORTA = 0) or (porta=5))  then PORTB:=0;  // Arrêt toutes les diodes sont éteintes
          if ((PORTA = 3) or (PORTA=6)) then PORTB:=$FF;  //    toutes les diodes sont allumées
         if ((PORTA = 1) or (PORTA=7)) then              //  clignotement des diodes
                        begin
                               PORTB:=$FF;
                               delay_ms(1000);
                               PORTB:=$00;
                              delay_ms(1000);
                         end;
                 if PORTA = 2 then       // Chenillard décalage à droite
                        begin
                                PORTB:=%10000000;
                                      for i:=1 to 8 do    //  Boucle pour
                                     begin
                                         delay_ms(50);
                                          PORTB:=PORTB shr 1 ;   // décalage à droite de 1 bit du PORTB
                                       end;
                         end;

                 if PORTA = 4 then       //  Chenillard décalage à gauche
                        begin
                                PORTB:=%00000001;
                                      for i:=1 to 8 do    //  Boucle pour
                                    begin
                                       delay_ms(50);
                                       PORTB:=PORTB shl 1 ;   // décalage à droite de 1 bit du PORTB
                                     end;
                         end;

          end;
        end.
```

ExerciceN°11 :

Feux tricolores de carrefour

Algorithme	Programme

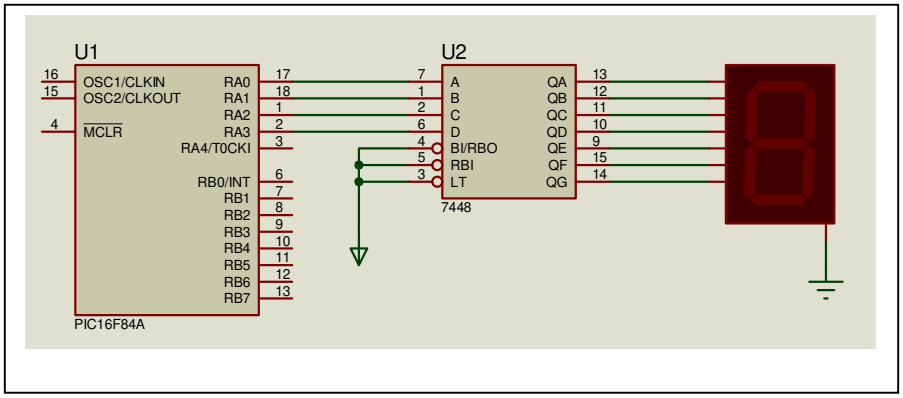
Algorithme : feu_tri
Variable :
S1 : bit du registre PORTA affecté à RA0 ;
Début
TrisA ⇐ $FF TrisB ⇐$00 PortB ⇐$00
Tant que (1=1) Faire
Début
SI S1=1 alors
Début
Portb ⇐ 1
Attente (60s)
Portb ⇐ 2
Attente (5s)
portb: ⇐ 4
Attente (55s)
Finsi
SI non
Début
portb: ⇐ 2
Attente (0,5s)
portb: ⇐ 0
Attente (0,5s)
Fin SI
Fin Faire
Fin
	program feu_tri;
	var
	S1:sbit at RA0_bit;
	begin
	trisA:=$FF;trisb:=$00;portb:=$00;
	while true do
	begin
	if S1=1 then
	begin
	Portb :=1 ;
	Vdelay_ms(60000);
	Portb :=2 ;
	delay_ms(5000);
	portb:=4;
	Vdelay_ms(55000)
	end
	else
	begin
	portb:=2;
	delay_ms(500);
	portb:=0;
	delay_ms(500);
	end;
	end;
	end.

Exercice N°12

Compteur modulo 10

1°) Algorithme d'un compteur modulo 10.

2°) Programme pascal correspondant.

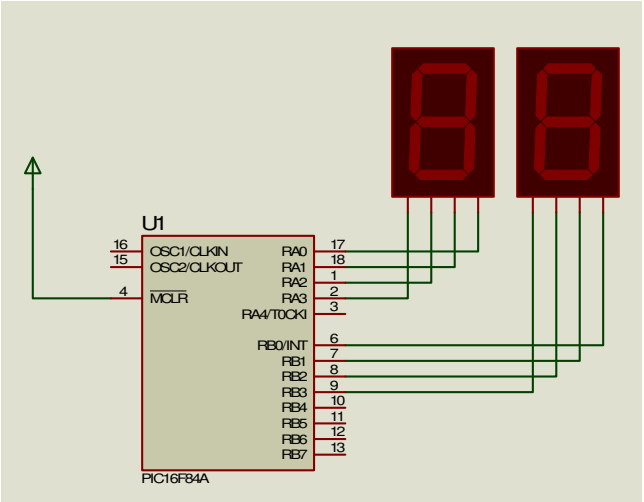


Algorithme	Programme

Algorithme : compteur m10
Variable i :entier
Début
TrisA ⇐ % 00000 PortA ⇐ % 00000
Tant que vrai faire
début
Pour i variant de 0 jusqu'à 9 faire
début
PORTA ⇐ i
Attente (0,5s)
Fin faire
Fin faire
Fin
	Program compteur m10 ;
	Var i:integer; // declaration d'une variable de type entier
	Begin
	TrisA := %00000 ; PortA:= %00000 ;
	while True do // boucle infinie
	begin
	For i:= 0 to 9 do // boucle répétitive
	begin
	PortA := i ;
	delay_ms(1000);
	end;
	end;
	End.

Exercice N°13

Compteur modulo 60



1°) Algorithme d'un compteur modulo 60

2°) Programme pascal correspondant.

Algorithme	Programme

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

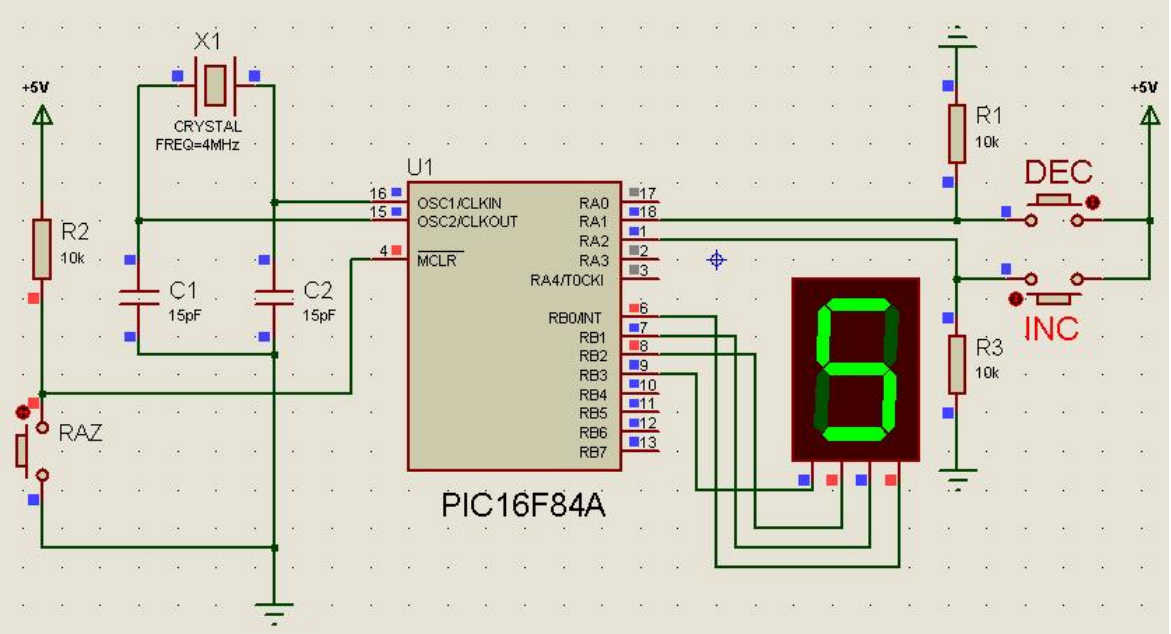
Exercice N°14

	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	bloqué	bloqué	oui	non	non	10ms
	bloqué	bloqué	bloqué	non	non	non	1ms
dizaine	bloqué	saturé	bloqué	non	oui	non	10ms
	bloqué	bloqué	bloqué	non	non	non	1ms
centaine	bloqué	bloqué	saturé	non	non	oui	10ms

2°)program exercice_N14_compteur_moduol1000_affichage_mutiplexe;

```
var i:word;
  var j:byte;
  var unite,dizaine,centaine:byte;
  begin
 trisb:=$F0;  // RB0,RB1,RB2,RB3 sorties RB4 à RB7 entrées
 trisa:=$18;    // RA0,RA1,RA2,sorties,RA3 et RA4 entrées
 porta:=0;    // initialiser le PORTA
 While true do     //  Boucle infinie
 begin
  for i:=0 to 999  do
  begin
   unite:= i mod 10;   //  identifier le chiffre de l'unité de i
   dizaine:= (i div 10) mod 10;  // identifier le chiffre de dizaine de i
   centaine:=  i div 100;   //  identifier le chiffre de centaine de i
 for j:=1 to 28 do
 begin
 porta:=%000;
 delay_ms(1);
 portb:=unite;
 porta:=%001;  // Commander le premier  afficheur
 delay_ms(10);
 porta:=%000;
 delay_ms(1);
 portb:=dizaine;
 porta:=%010;     // Commander le 2éme afficheur
  delay_ms(10);
 porta:=%000;
 delay_ms(1);
 portb:=centaine;
 porta:=%100;
 delay_ms(10);
  end;
 end;
 end;
end.
```

ExerciceN°15:



Programme

```
program exercice_N_15_button;

var x:byte;

begin

 trisa:=$1F;trisb:=0;portb:=0;x:=0;

 while true do

 begin

  if button(porta,2,100,1) then INC(x);    // ou  x:=x+1;

  if x=10 then x:=0;

   if button(porta,1,100,1) then DEC(x);     // ou  x:=x-1;

  if x=255 then x:=9;

  portb:= x;

  end;

  end.
```

Exercice N°16:

1°) Configuration du registre INTCON :

INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Valeur INTCON
	1	0	0	1	0	0	0	0	$(90)_{16}$

2°) Programme

Programme

```
program exercice_N_16_interrupt;
 VAR i:byte;
procedure interrupt;
  begin
  intcon :=$90; i:=i+1 ;
  end;
   begin
 trisa:=0;porta:=0; intcon :=$90; i:=0;
    while(1=1) do
      begin
        porta:=i;
        if i=10  then i:=0;
        end;
    end.
```

Exercice N°17:

Programme

Programme

```
program exercice_comp_decomp_modulo7_interruption;
var a: sbit at RB1_bit;
var x: byte;
procedure interrupt;
   begin
   if (a=1) then x:=x-1
   else x:= x+1
   intcon.INTF :=0  ;  // Remise à zéro de l'indicateur INTF
   intcon.GIE :=1   ;       // Réactivation globale des interruptions
   end ;
begin
intcon := $90;
trisA := $00;
trisB := $FF;
x:=$00;
 while (1=1) do  // boucle infinie
  begin
   portA := x;
   if  (x = $FF) then x:=$06
   else if (x = $07) then x:=$00 ;
 end;
end.
```

Exercice N°18

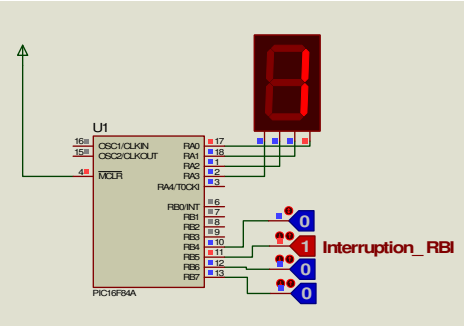
Réaliser un compteur modulo 8 sachant qu'a chaque changement d'état sur au moins une des entrées RB4 à RB7 du PORTB le compteur s'incrémente : (Utiliser la procédure d'interruption externe avec RBI)

1°) Configuration du registre INTCON :

INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Valeur INTCON
	1	0	0	0	1	0	0	0	$(88)_{16}$

Programme

```
program exercice_N_18_interrupt;
  VAR X,etat:byte;
procedure interrupt;// lecture du portb pour déverrouiller l'accès au bit RBIF
  begin
  etat:=PORTB;
  x:=x+1 ;
  intcon :=$88;
  end;
   begin
 trisa:=0; Trisb:=$FF;porta:=0;X:=0;
 intcon :=$88;
    while(1=1) do
      begin
        porta:=x;
        if x=8 then x:=0;
        end;
    end.
```



Exercice N°19

1°) Configuration du registre INTCON :

INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Valeur INTCON
	1	0	0	1	1	0	0	0	$(98)_{16}$

Programme

```
program exercice_N19_2interruption;
var
  etat:byte;
  X:byte ;
  procedure interrupt; // Procédure d'interruption
    begin
   if INTCON.intF = 1 then
  begin
     inc(X);
     intcon:=%10011000;
     end;
  if INTCON. RBIF = 1 then
  begin
     etat:=portb;
     dec(X);
     intcon:=%10011000;
   end;
end;
begin
  TRISC:=0;
  TRISB:=$FF;
  PORTA:=0;
  intcon:=%10011000;
  while true do
  begin
   PORTC:=X
   end;
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 mode de fonctionnement du TMR0 est compteur ou temporisateur :Compteur

2°) Donner alors le nom de la broche de l'entrée d'horloge du TMR0 : RA4

3°) 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**	**0**	**1**	**0**	**0**	**0**

2°) Compléter le programme suivant :

```
program exercice_N_20_TIMER0_comp9;
begin
   TRISB:=$ 00;
   TRISA:=$ 1F ;
   OPTION_reg:= %11101000;
   TMR0:=0;
   while true do
   begin
     portb:=TMR0;
     if TMR0=9 then TMR0:=0;
     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**	**1**	**1**	**0**	**0**	**0**	**0**

2°) Compléter le programme suivant :

```
program exercice_N_21_TIMER0_comp9;
begin
   TRISB:=$F0; // Configration PORTB
   TRISA:=$1F ; // Configration PORTA
   OPTION_reg:= %11110000   ;
   TMR0:=0; // Initialisation du TMR0
   while true do
   begin
      portb:=TMR0;
      if TMR0=16 then TMR0:=0;
      end;
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.

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**	**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	bloqué	oui	non	10ms
	bloqué	bloqué	non	non	1ms
Dizaine de TMR0	bloqué	saturé	non	oui	10ms
	bloqué	bloqué	non	non	1ms

3°) Programme :

```
program Exercice_22_TIMER0_comp_100;

Var
 uni:byte;
 dix:byte;


begin
trisb:=$F0;            // de RB0 à RB3  sorties ,de RB4 à RB7 entrées
trisa:=$1C;            //  de RA0 et RA1 sorties ,RA2 à  RA4 entrées


TMR0:=0;       // initialisation du timer 0 à la valeur 0
OPTION_REG := %11100000;
while true do
begin


      While TMR0 < 100 do


       begin
    uni  := TMR0 mod 10;              // Identifier le chiffre de l'unité du TIMER0
```

```
dix  := TMR0 div 10 ;            // Identifier le chiffre de dizaine de la variable i

        porta:=0;
        delay_ms(1);
        portb:=uni;
        porta:=1;
        delay_ms(10);            //  affichage multiplexé puisqu'on dispose d'un seul décodeur
        porta:=0 ;
        delay_ms(1);
        portb:=dix;
        porta:=2;
        delay_ms(10);

     end;
TMR0:=0;
end;
end.
```

Exercice N°23

Ue ⟶ N

5V ⟶ 1023

$$N = \frac{1023}{5} Ue$$

2°) Configurer les entrées /sorties :

TRISA

1	1	1	1	1	1

TRISB

0	0	0	0	0	0	0	0

TRISC.0= 0 TRISC.1= 0

3°) Configurer le registre ADCON 1

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**

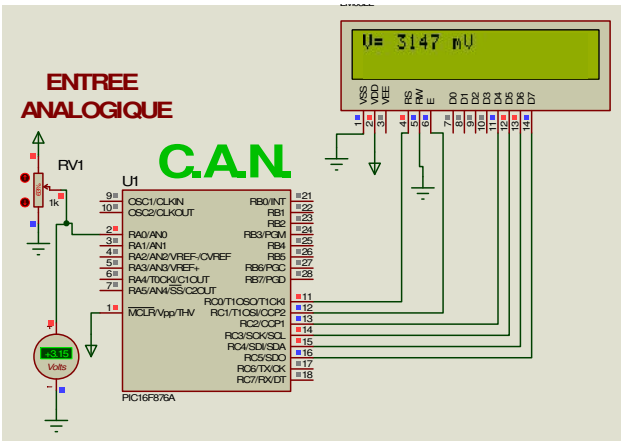
4°) Programme

```
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 : sbit at RC3_bit;
var LCD_D4 : sbit at RC4_bit;
var LCD_D5 : sbit at RC5_bit;
var LCD_D6 : sbit at RC6_bit;
var LCD_D7 : sbit at RC7_bit;

var LCD_RS_Direction : sbit at TRISC2_bit;
var LCD_EN_Direction : sbit at TRISC3_bit;
var LCD_D4_Direction : sbit at TRISC4_bit;
var LCD_D5_Direction : sbit at TRISC5_bit;
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:=%10000000;    // justification des 10 bits a droite et RA2 entrée ANA
   TRISA  := $FF;         // PORTA Entrées
   TRISB  := 0;           // PORTB Sorties
   TRISC.0  := 0;          // la broche RC0 est cofigurée comme sortie
   TRISC.1  := 0;          //  la broche RC1 est cofigurée comme sortie
   Lcd_out(1,1,'conversion est:'); // Ecrire « conversion est : » à la première ligne et premier colonne
   while (TRUE) do
     begin
       N := ADC_Read(2); // lecture de la valeur lue par le convertisseur sur le canal 3

       PORTB := N;  // Les 8 bits de plus faibles poids sont aux PORTB
       PORTC := N shr(8);  // 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(2,5,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

Analogique

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

TRISA

1	1	1	1	1	1

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	0	0	0	0	0	B9	B8

ADRESL

B7	B6	B5	B4	B3	B2	B1	B0

10 bits du résultat

3°) Configurer le registre ADCON 1

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
1	**0**	**0**	**0**	**0**	**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 :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
  TRISA:=$3F;
  ADCON1:=%10000000;
  LCD_init();
  LCD_cmd(_LCD_cursor_off);
  LCD_out (1,1,'V=');
   while true do
   begin
        N:= ADC_read(0);  // Lecture de la conversion sur le canal 0
        calcul:= (N*(5000/1023));
        tension := word(calcul);   // forcage de type transformation en mot (partie entière)
        wordtostr(tension,valeur_affichage); // transformation de la tension en texte
        LCD_out (1,3,valeur_affichage);
        LCD_out (1,9,'mV');  // Afficher au premier ligne et 9ème colonne 'mV'
        delay_ms(1000);
        end;
end.
```

Exercice N°25 :

```
program exercice_N_25_MLI;
begin
PWM1_init(1000); // initialisation du PWM à 500Hz
PWM1_start;
while true do
begin
PWM1_Set_Duty(127);
end;
end.
```

Exercice N°26 :

Entrées	Rapport cyclique	N
PORTB =0	$\alpha = 0$	N = 0
PORTB =1	$\alpha = 0,25$	N = 64
PORTB =3	$\alpha = 0,5$	N =127
PORTB =7	$\alpha = 0,75$	N =192
PORTB =15	$\alpha = 1$	N =255

```
program exercice_N_26_MLI;

begin

  TRISB:=$FF;

  PWM1_Init(1000);

  PWM1_Start;

  while true do

  begin

  if PORTB=0 then PWM1_Set_duty(0);

  if PORTB=1 then PWM1_Set_duty(64);

  if PORTB= 3 then PWM1_Set_duty(127);

  if PORTB= 7 then  PWM1_Set_duty(192);

  if PORTB= 15 then PWM1_Set_duty(255);

  end;

end.
```

Exercice N°27 :

1°) Si on appui sur le bouton RA2 la vitesse du moteur : augmente puisque le rapport cyclique augmente

Si on appui sur le bouton RA1 la vitesse du moteur : diminue puisque le rapport cyclique diminue

2°) Compléter le programme :

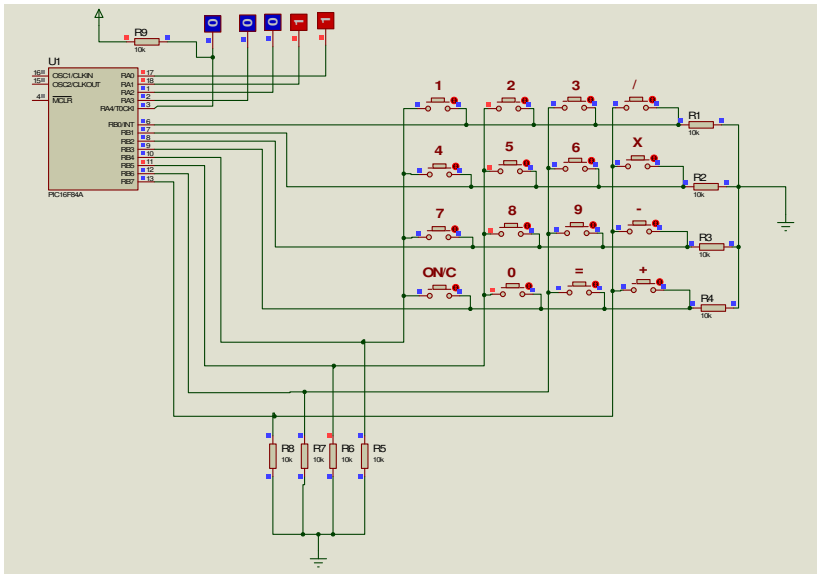
```
program exercice_N_27_MLI;
var x:byte;
begin
 PWM1_Init(1000);
 PWM1_Start();
ADCON1:=$07;  // PORTA numérique
 trisa:=$3F;trisb:=0;portb:=0;x:=0;
 while true do
 begin
  if button(porta,2,100,1) then INC(x); if x=255 then dec(x);
  if button(porta,1,100,1) then DEC(x); if x=0 then inc(x);
  portb:= x;
  PWM1_Set_duty(x);
  end;
  end.
```

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

Si on appui sur le bouton RA2 on incrémente la variable x « de type octet » si la variable atteint sa valeur maximale 255 alors on le décrémente pour ne pas avoir la variation brusque de la vitesse maximale à l'arrêt

Si on appui sur le bouton RA1 on décrémente la variable x « de type octet » si la variable atteint sa valeur minimale 0 alors on l'incrémente pour ne pas avoir la variation brusque de l'arrêt à la vitesse maximale

Exercice N°28 :



Touche	Kp	PortA
1	1	1
2	5	5
3	9	9
4	2	2
5	6	6
6	10	10
7	3	3
8	7	7
9	11	11
0	8	8
ON/C	4	4
+	16	16
/	13	13
=	12	12
-	15	15
x	14	14

Tableau 1

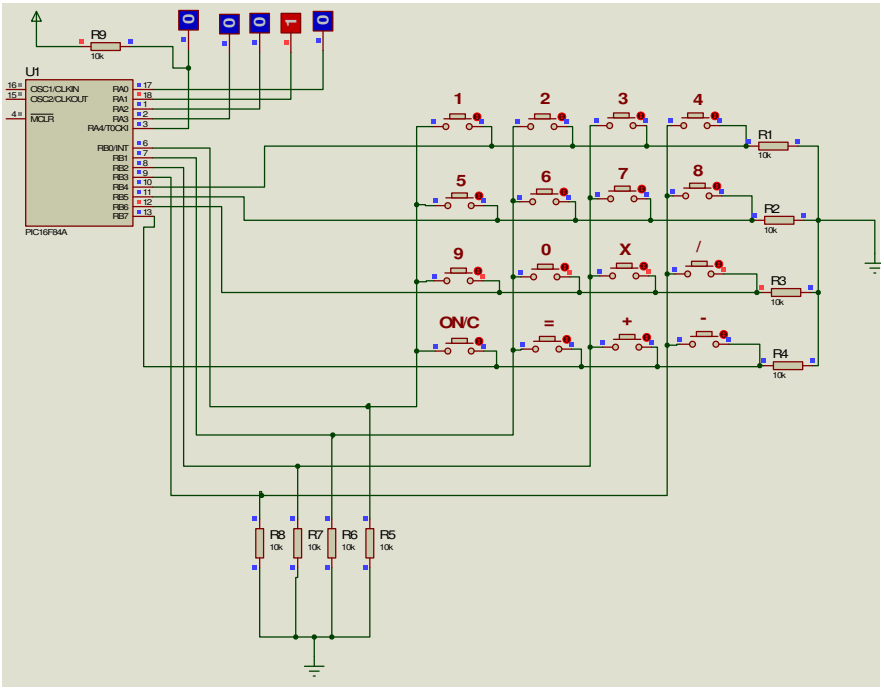


Tableau 2

Touche	Kp	PortA
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
0	10	10
ON/C	13	13
+	15	15
/	12	12
=	14	14
-	16	16
x	11	11

Montage 3

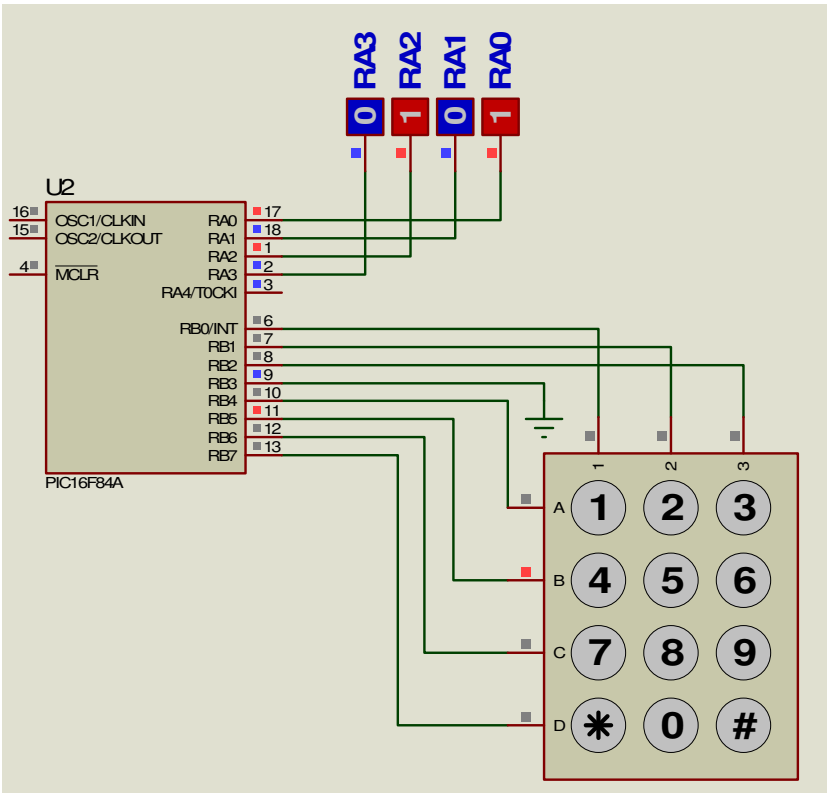


Tableau 3

Touche	Kp	PortA
1	1	1
2	2	2
3	3	3
4	5	5
5	6	6
6	7	7
7	9	9
8	10	10
9	11	11
*	13	13
0	14	14
#	15	15