

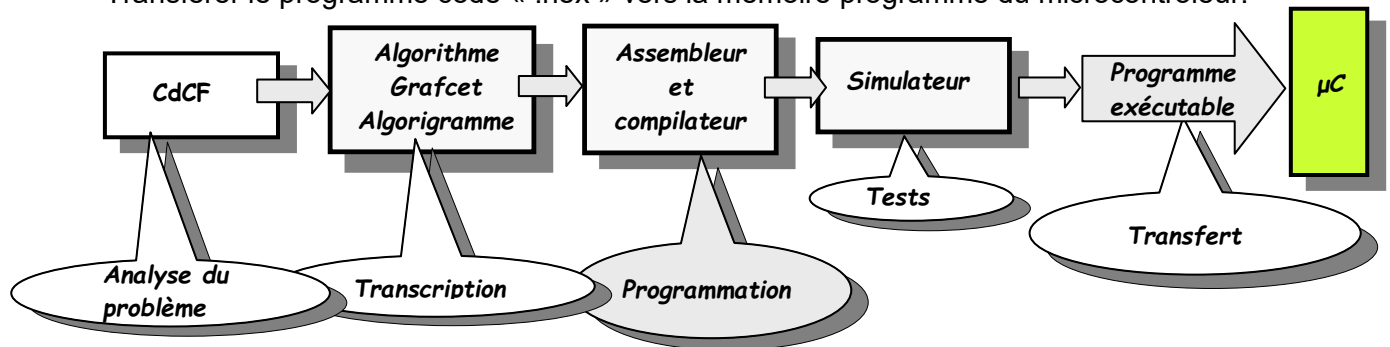
MISE EN OEUVRE D'UNE APPLICATION A BASE DE MICROCONTROLEURS

Objectifs de la leçon : - Traduire un cahier des charges en un programme en utilisant un langage évolué
- Simuler un programme
- Transférer un programme vers un microcontrôleur

A) - Démarche de la mise en œuvre d'une application :

La mise en œuvre d'une application, à base de microcontrôleur, consiste à :

- Traduire un cahier des charges en un algorithme ou algorithme.
- Ecrire un programme mettant en œuvre un langage évolué tels que langage C, Basic, Pascal, etc...
- Assembler et compiler le programme (avoir un programme code d'extension .hex)
- Simuler le programme (exemple avec ISIS ou Flowcode)
- Transférer le programme code « .hex » vers la mémoire programme du microcontrôleur.



Divers outils de développement sont mis à la disposition du concepteur, parmi ces outils on cite :

- la programmation mettant en œuvre un langage évolué tels que : langage C, Basic, Pascal...
- La programmation graphique. Elle est basée sur l'interconnexion graphique de symboles ou modules (algorithme, GRAFCET ...)

Remarques :

- La suite de cette leçon portera sur le langage C

L'algorithme :

Définition : c'est une série des actes ou opérations élémentaires, qu'il faut exécuter en séquence pour accomplir une tâche quelconque, en suivant un enchaînement strict.

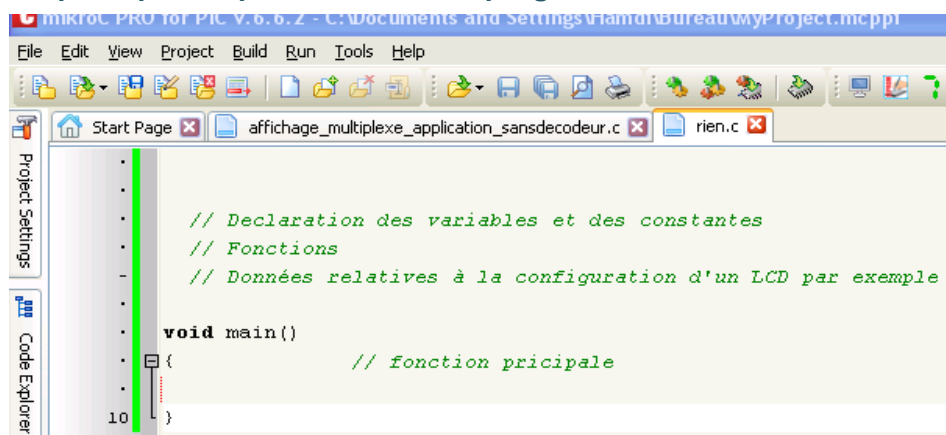
Représentation normalisée : l'algorithme s'établit par une succession de phrases simples.

1- Introduction au langage de programmation mikroC :

a) Structure d'un programme en mikroC :

Un programme sous MikroC PRO peut être décomposé en deux grandes parties ou zones : une zone réservée aux différentes déclarations et une autre zone réservée au programme principale void main ()

Main : Fonction principale et point d'entrée du programme.



b) Les règles de bases :

- Les instructions propres au langage mikroC doivent être écrites en **minuscule** (exemple : void main)
- Les instructions relatives aux microcontrôleurs sont généralement écrites en **majuscule** (exemple : TRISB) mais le compilateur les accepte même si elles sont écrites en minuscule.
- Les retours à la ligne et les espaces servent uniquement à aérer le code.
- Toutes instructions ou actions se terminent par un point virgule « ; ».
- Un bloc d'instruction commence par une accolade ouverte « { » et se termine par une accolade fermée « } »

c) Quelques types de variables utilisées en MikroC :

Type	Taille en octet	Longueur en bit	Rang
bit	x	1bit	0 ou 1
sbit	x	1bit	0 ou 1
(unsigned)char	1	8bits	0 à 255=2 ⁸ - 1
signed char	1	8bit	-2 ⁷ = -128 à 127=2 ⁷ - 1
(signed) short	1	8bits	-128 à 127
(signed) int	2	16bits	-2 ¹⁵ = -32768 à +32767=2 ¹⁵ - 1
unsigned (int)	2	16bits	0 à +65535 = 2 ¹⁶ - 1
(signed) long (int)	4	32bit	-2 ³¹ à +2 ³¹ - 1
unsigned long (int)	4	32bit	0 à +2 ³² - 1
float	4	32bits	-1.5 * 10 ⁴⁵ à +3.5 * 10 ³⁸

d) Les bases du compilateur MikroC : pour le même nombre N on peut le coder en :

- ☞ **Binaire (base 2) :** on doit commencer par **0b...** ou **0B ...** Exemple : N= 0b1101010
- ☞ **Décimale (base 10) :** on écrit directement le nombre Exemple : N= 106
- ☞ **Hexadécimale (base 16) :** on doit commencer par **0x....** ou **0X...** Exemple : N= 0x6A
- ☞ **octale (base 8) :** commence par **0** Exemple : N= 0152

❖ Les opérateurs logiques (appliquer pour des variables à un seul bit)

Une variable booléenne est une variable pouvant prendre la valeur *vrai* ou *faux* (0 ou 1).|

La valeur d'une expression booléenne est, comme le résultat des comparaisons, une valeur entière.

e) les opérateurs en mikroC :

❖ Les opérateurs arithmétiques

Opérateurs arithmétiques en mikroC		
Opérateur	opération	Notation
+	Addition	x+y
-	Soustraction	x-y
*	Multiplication	x*y
/	Division	x/y
%	modulo	x%y

Opérateur	Nom	Notation
&&	ET	X&&y
	OU	x y
! (unaire)	Non	!x

Les opérateurs de manipulation de bits (appliquer pour des variables à un ou plusieurs bits)

Opérateur	Nom	Notation
&	ET bit à bit	x & y
	OU bit à bit	x y
^	OU exclusif bit à bit	X^y
~ (unaire)	NON bit à bit	~x
>>	Décalage à droite	x>>2
<<	Décalage à gauche	X<<3

Opérateur avec Alt Gr 6

~ Opérateur avec Alt Gr 2 puis espace

x est décalé 2 fois à droite

x est décalé 3 fois à gauche

❖ Opérateurs unaires d'incrémentement ou de décrémentation

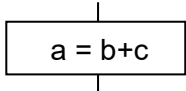
Opérateur	Notation	Équivalent
++	x++ ou ++x	x=x+1
--	x-- ou --x	x=x-1

❖ **Les opérateurs de comparaison :**

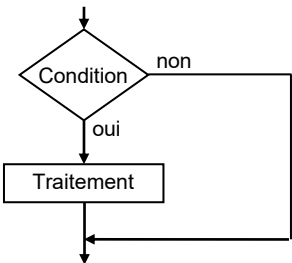
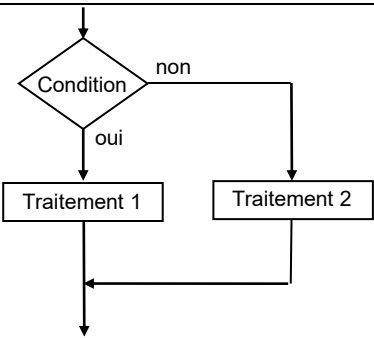
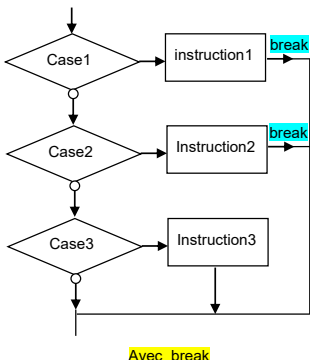
Opérateur	Nom	Notation
==	Test d'égalité	$x==y$
!=	Test de non égalité	$x!=y$
<=	Test inférieur ou égal	$x<=y$
>=	Test supérieur ou égal	$x>=y$
<	Test inférieur strict	$x<y$
>	Test supérieur strict	$x>y$

f) **Les structures usuelles :**

❖ **L'affectation :** C'est l'action d'attribuer une valeur à une variable.

Langage graphique	Langage algorithmique	Langage C
	$a \leftarrow b+c$	$a = b+c$

❖ **Les structures alternatives :**

Langage graphique	Langage algorithmique	Langage C
	SI (condition) FAIRE DEBUT traitement FIN ;	if (condition) { traitement ; }
	SI (condition) FAIRE DEBUT traitement 1 FIN SINON DEBUT traitement 2 FIN SINON	if (condition) { Traitement 1 ; } else { traitement 2 ; }
	SELON (expression) DEBUT Valeur 1 : instruction1 Valeur 2 : instruction2 Valeur n : instruction n default : instruction 0 FINSELON	switch (expression) { case 1 : instruction1 ; break ; case 2 : instruction2 ; break ; case N : instruction N ; break ; default : instruction 0 ; }

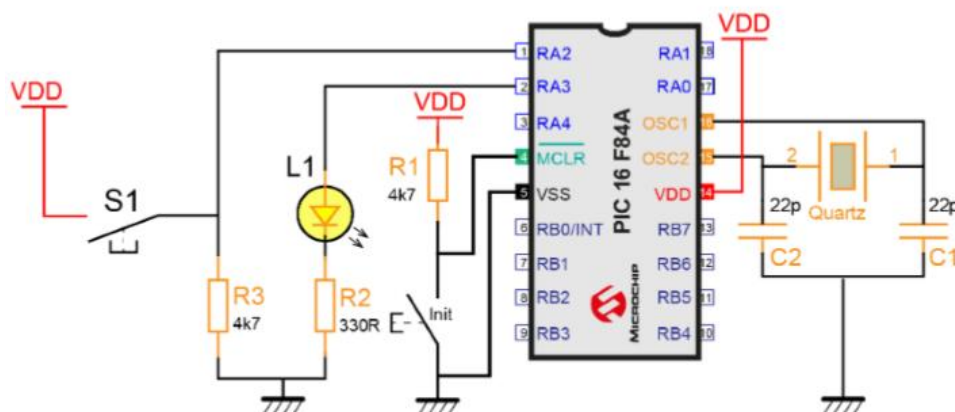
❖ Les structures itératives ou répétitives :

Langage graphique	Langage algorithmique	Langage PASCAL
	i : entier DEBUT POUR i variant de « valeur initiale » à « la valeur finale » FAIRE DEBUT Instructions FINFAIRE FIN POUR	int i ; for (i=0 ;i<10 ;i++) { Instructions; }
	TANQUE (condition) FAIRE DEBUT Action ; FIN TANT QUE	while (condition) { Action ; }
	REPETER DEBUT Action ; FIN ; JUSQU'A Condition	do { Traitement ; } while (Condition) ;

4) Applications :

☺ Activité N°11:

Le fonctionnement de la diode LED (L1) dépend des états du bouton-poussoir (S1) :
Si S1= 1 alors L1= 0 si non L1=1.



Compléter le programme du fonctionnement de la diode L1 puis vérifier le fonctionnement.

NB : Les broches non utilisées sont configurées comme des entrées

Algorithme

S1 type bit $\rightarrow$ RA2
L1 type bit $\rightarrow$ RA3

DEBUT

TRIS A $\leftarrow$ (11110111)₂
L1 $\leftarrow$ 0

TANT QUE
DEBUT

SI (S1=1) Alors faire

L1 $\leftarrow$ 0

SINON FAIRE

L1 $\leftarrow$ 1

FIN TANT QUE

FIN Programme

Programme

```
sbit S1 at RA2_bit;
sbit L1 at RA3_bit;

void main()
{
    TRISA= 0b11110111;
    L1 = 0;
    while (1)
    {
        if ( S1==1 )
        {
            L1=0;
        }
        else
        {
            L1=1;
        }
    }
}
```

N.B : $\leftarrow$: affecter une valeur à une variable ; $\rightarrow$: relier une variable à une broche.

☺ **Activité N°12 :** Réaliser l'activité N° 3 du manuel d'activités page 139 jusqu'à 142

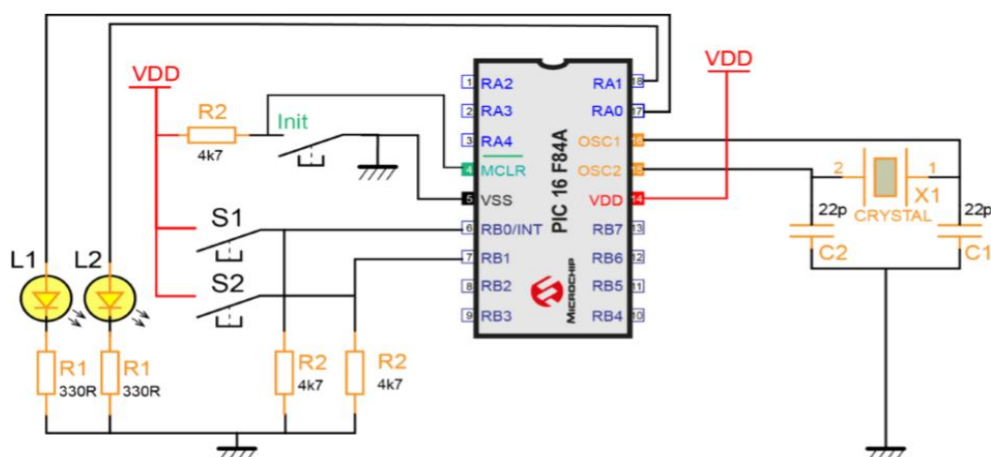
☺ **Activité N°13:**

Le fonctionnement de deux diodes LEDs (L1 et L2) dépend des états respectifs des boutons poussoirs (S1 et S2) :

➤ Dans le cas où les états de S1 et S2 sont différents on aura L1=1 et L2=0 si non L1 = 0 et L2=1.

Compléter puis saisir le programme suivant puis simuler et vérifier le fonctionnement :

N.B: Les broches non connectées sont configurées en entrées



Algorithme

Fonction principale

DEBUT

TRISA $\leftarrow (11100)_2$

RA0 $\leftarrow 0$

RA1 $\leftarrow 0$

TRISB $\leftarrow (11111111)_2$

TANT QUE

DEBUT

SELON (PORTB)

DEBUT

Cas où PORTB= 1

RA0 $\leftarrow 1$, RA1 $\leftarrow 0$; sauter ;

Cas où PORTB= 2

RA0 $\leftarrow 1$, RA1 $\leftarrow 0$; sauter ;

Défaut

RA0 $\leftarrow 0$, RA1 $\leftarrow 1$

FIN SELON

FIN TANT QUE

FIN fonction principale

Programme

Void main ()

{

TRISA = 0b11100 ;

PORTA.RA0 = 0 ;

PORTA.F0 = 0 ;

TRISB = 0b 11111111 ;

while(1)

{

switch(PORTB)

{

case 1:

PORTA.RA0= 1 ; PORTA.RA1= 0 ; break ;

case 2:

PORTA.RA0= 1 ; PORTA.RA1= 0 ; break ;

default:

PORTA.RA0=0 ; PORTA.RA1= 1 ;

}

}

}

😊 Activité N°14:

On donne le programme en mikroC suivant :

sbit D at RA0_bit;

char i ; // déclaration d'une variable de type entier codé sur un octet

void main()

{

TRISA.RA0=0; // broche RA0 sortie

PORTA=0;

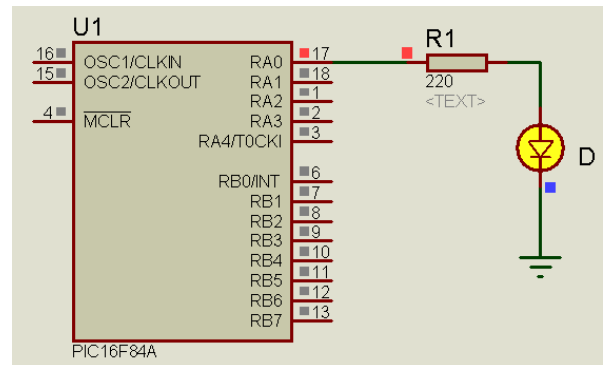
for (i=0;i<5;i++)

{

D=1;delay_ms(500);D=0;delay_ms(500);

}

}



Expliquer le fonctionnement de la diode LED D :

La diode LED D clignote à la fréquence 1 Hz 5 fois puis elle reste éteinte.

Dans la boucle **for** le programme exécute le clignotement de la diode pour la première fois pour i=0, puis une deuxième fois pour i=1, la troisième fois pour i=2, la quatrième pour i=3 et enfin la cinquième pour i=4 et le programme sort de la boucle **for** pour i=5 et se met à sa fin.

Remarque :

Il est possible d'accéder individuellement à 1 seul bit du PORT. A titre d'exemple, on écrit :

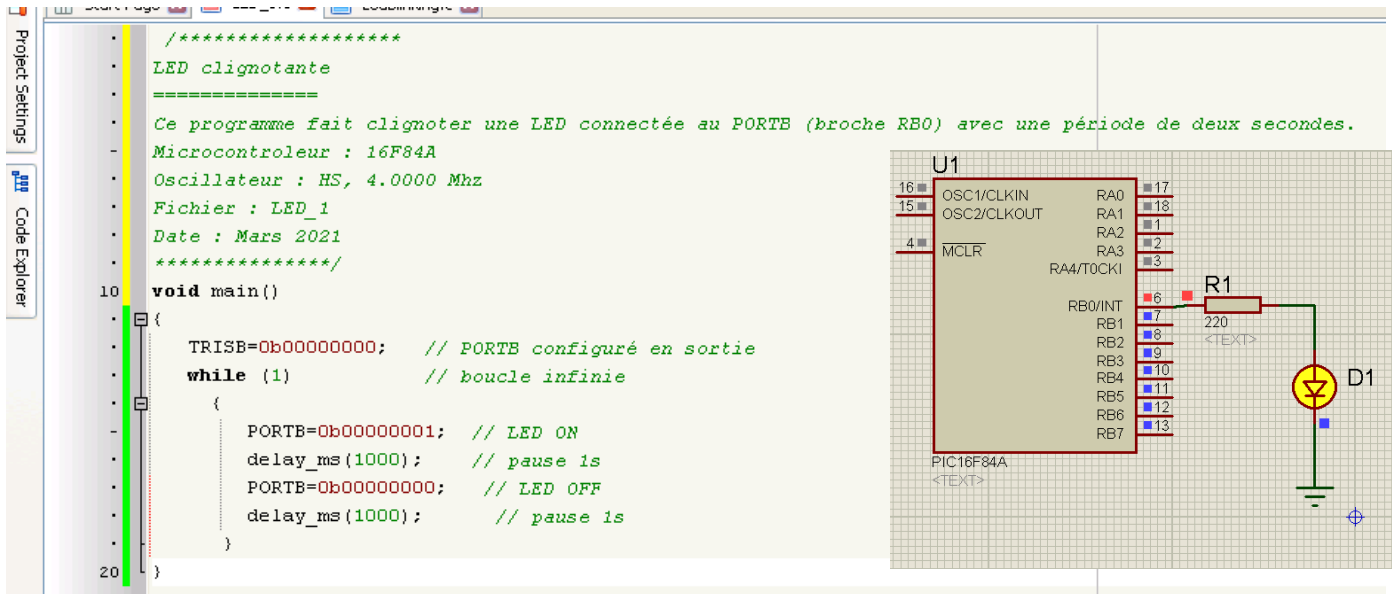
`TRISB.F0 = 0` ; ou `TRISB.B0=0` ; ou `TRISB.RB0` ou `TRISB2_bit`

`PORTB.F0 = 1` ; ou `PORTB.B0=1` ; ou `PORTB.RB0=1` ; ou `RB0_bit=1` ;

Exemple :

On donne ci-dessous un exemple simple d'un programme en mikroC. Ce programme assure le clignotement d'une diode Led une fois par seconde (1 seconde allumée et une seconde éteinte).

Version 1 :



Version 2 :

/* ***** LED clignotante

Ce programme fait clignoter une LED connectée au PORTB (broche RB0) avec une période de deux secondes.

Microcontrôleur : 16F84A

Oscillateur : HS, 4.0000 Mhz

Fichier : LED_1

Date : Mars 2021

Les commentaires sont utilisés pour préciser le fonctionnement du programme.

Les lignes de commentaires sont ignorées et non compilé par le compilateur.

Long commentaire commence par /* et se termine par */

Court commentaire sur une ligne commence par le caractère « // »

***** */

void main()

En langage C, un programme commence avec les mots clés : Void main ()

TRISB = 0; // Configuration du PORTB en sortie

PORTB=0 ; // Initialisation du PORTB

for(;;) // Boucle sans fin

Après cela, une accolade ouvrante est utilisée pour indiquer le début du corps de programme au lieu de begin

{ PORTB.B0 = 0; // RB0 = 0

Delay_ms(1000); // Pause d'une seconde

PORTB.B0 = 1; // RB0 = 1

Delay_ms(1000); // Pause d'une seconde

Le programme se termine par une accolade fermante Au lieu de end.

} // Fin de la boucle



Le schéma d'une carte électronique (figure 3) illustrant l'installation d'éclairage de la maison (figure 1) :

- saisir sur le logiciel "Proteus ISIS" ce schéma ;

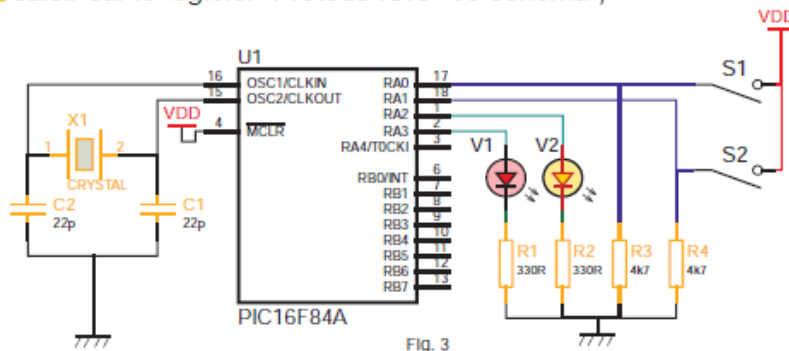
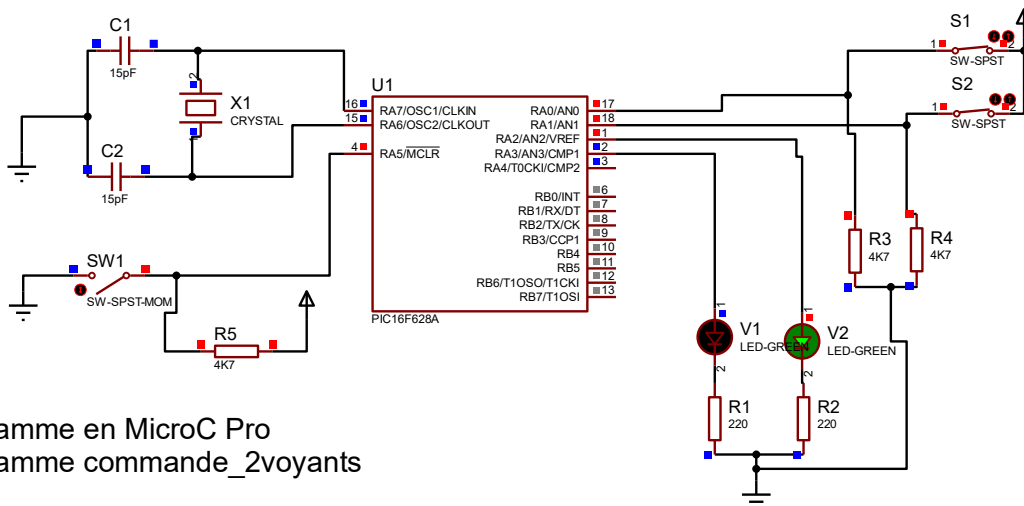


Fig. 3

Table de vérité :

S1	S2	V1	V2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

schémas sur le logiciel de simulation ISIS



Programme en MicroC Pro
Programme commande_2voyants

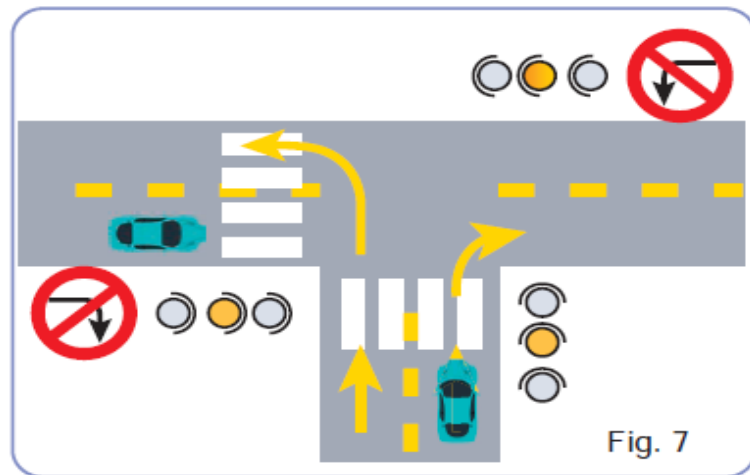
```

sbit S1 at RA0_bit;
sbit S2 at RA1_bit;
sbit V1 at RA2_bit;
sbit V2 at RA3_bit;

void main() {
    TRISA=0X13; PORTA.B2=0;PORTA.B3 =0;
    Cmcon=0x07 ;
    while (1) {
        if(S1==0 && S2==0){V1=0;V2=0;} else
        if(S1==0 && S2==1){V1=1;V2=1;} else
        if(S1==1 && S2==0){V1=1;V2=1;} else
        {V1=0;V2=1;}
    }
}
    
```

Activité3 : feux de carrefour

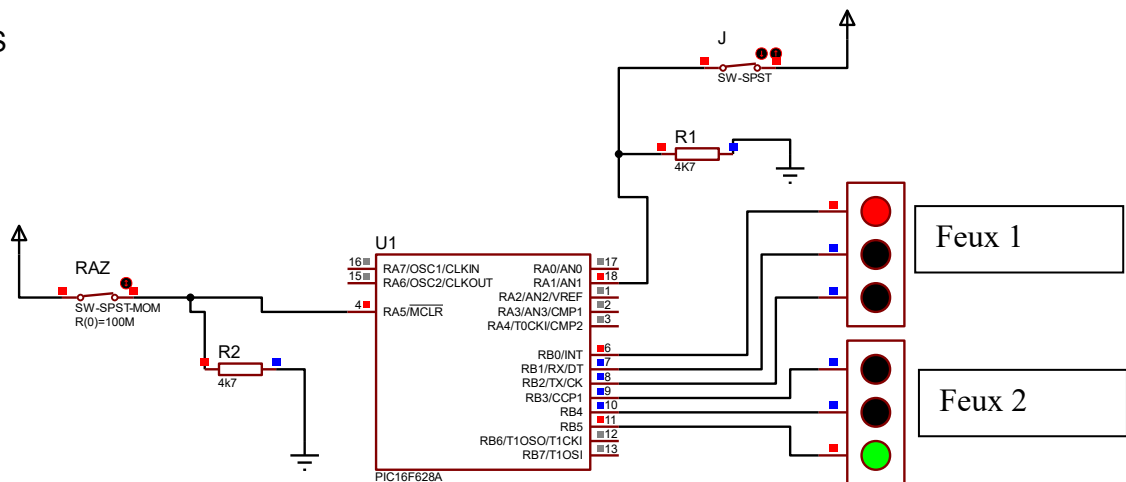
La figure 7 illustre un projet, réalisé par une municipalité, destiné à l'organisation de la circulation de voitures au niveau d'un croisement d'une route principale avec une route secondaire.



Le cahier des charges fonctionnel de ce projet exige que l'organisation de la circulation assurée par un feu tricolore fonctionnant en deux modes (figure

- Mode jour utilisant les trois feux fonctionnant en alternance à des durées fixes ; J=1
- Mode nuit utilisant uniquement le feu jaune clignotant ; J=0

Schéma sur ISIS



Programme en MiKroC Pro:

```
# define J1 RA1_bit
```

```
void main() {
    TrisA=0XFF; TrisB=0X00;
    PortB=0 ;CMCON=0x07;
    while(1)
    {
        if(J1==1) {portB=0x21; delay_ms(5000); PortB=0X0B;
        delay_ms(2000); PortB=0X0C;
        Delay_mS(7000);
        PortB=0X19; delay_ms(2000);}
        else {PortB=0X12; delay_ms(500); portB=0; delay_ms(500);}
    }
}
```

Séquence de fonctionnement : (pour mode Jour)

Feux1 :V=0 ; J=0 ; R=1 -- Feux2 : V=1 ; J=0 ; R=0 ;
 Pause 7 s
 Feux1 :V=0 ; J=1 ; R=1 -- Feux2 : V=0 ; J=0 ; R=1 ;
 Pause 2 s
 Feux1 :V=1 ; J=0 ; R=0 -- Feux2 : V=0 ; J=0 ; R=1 ;
 Pause 5 s
 Feux1 :V=0 ; J=0 ; R=1 -- Feux2 : V=0 ; J=1 ; R=1 ;
 Pause 2 s