Université des Sciences et de la Technologie d'Oran –Mohamed Boudiaf–

Faculté de Génie Electrique

Département d'Electronique

Licence 3ème Année Electronique

Travaux Avant Projet

Polycopié des TP

de la Programmation Graphique du Microcontrôleur PIC16F877A avec le Logiciel Flowcode

Fait Par :

Mr Daoud A.

Année Universitaire 2015/2016

Préface

Ce polycopié de TP regroupe un certain nombre d'applications au tour de la programmation du microcontrôleur PIC16F877A de Microchip. Les exemples cités dans ce polycopié doivent fournir l'opportunité d'apprendre la programmation en manipulant des organigrammes par la méthode graphique qu'offre le logiciel 'Flowcode for PICmicros'. Ainsi, l'objectif final est de simplifier la phase programmation dans le but d'apprendre à faire des montages expérimentaux à base de PIC sans se heurter à la lourdeur de la programmation en assembleur (MPLAB) ou en langage évolué (langage C (compilateurs : CCS C, CC5X, MikroC, HI-TECH C), langage PIC Basic, MikroPascal,.. etc).

La première partie de ce polycopié est réservée à la présentation de l'interface du logiciel Flowcode alors que la deuxième partie inclut plusieurs exemples dont les organigrammes sont simples et peuvent être reproduits par des étudiants n'ayant pas une connaissance de base en programmation des PICs.

Tous les exemples cités dans ce polycopié ont été testés avec les logiciels de simulation Flowcode V5 et Proteus ISIS V7.9. La validation à été faite avec les kits easyPIC6 et easyPIC2.

Sommaire

1- Introduction

Flowcode V5 for PICmicros est un logiciel de programmation graphique. Il permet de créer des applications pour des microcontrôleurs en sélectionnant et plaçant des icônes sur un environnement afin de créer des programmes simples (Organigrammes). Ces programmes peuvent contrôler des périphériques externes connectés au microcontrôleur comme les LED, un afficheur LCD etc. Flowcode permet de simuler cet organigramme avant de le compiler et de l'assembler et de le transférer vers le microcontrôleur sous forme d'un fichier binaire (code objet). Pour atteindre cet objectif, il suffit de réaliser les étapes suivantes :

a) Créer un nouvel organigramme en spécifiant le type de microcontrôleur cible.
b) Sélectionner et faire glisser les icônes de la barre des composants vers l'organigramme.
c) Ajouter les périphériques externes nécessaire en cliquant sur les boutons correspondants dans la barre d'outils des composants, éditer leurs propriétés, spécifier comment ils sont connectés au microcontrôleur et appeler les macros associées aux périphériques.
d) Lancer la simulation afin de s'assurer que l'application (Organigramme) fonctionne correctement.
e) Transférer l'application vers le microcontrôleur par compilation de l'organigramme vers le langage C et ensuite vers du code objet.

2- Les Barres d'outils

Flowcode permet de représenter le programme à implémenter dans un PIC microcontrôleur sous forme d'un organigramme. La figure 1 présente une copie d'écran de l'environnement de travail du logiciel et ses principales fonctions.

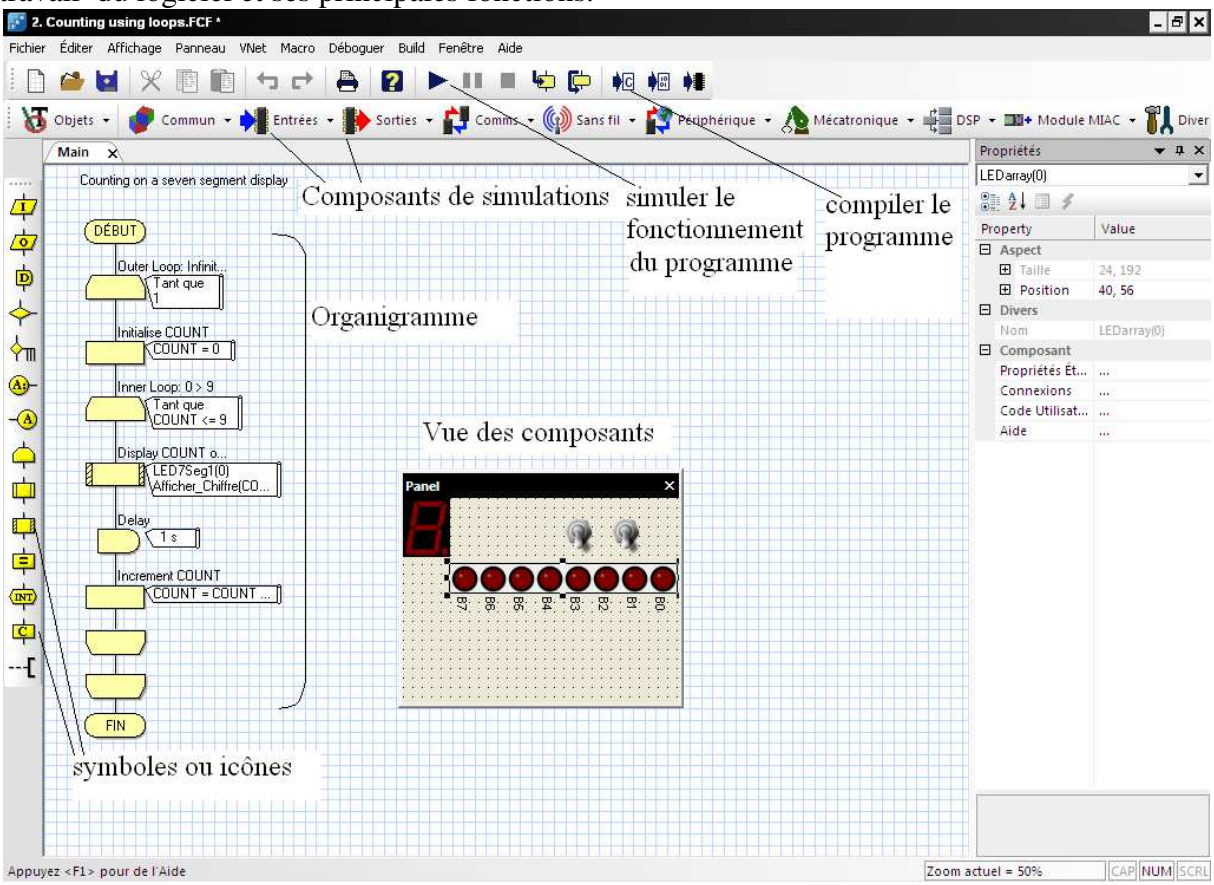


Figure 1 : Environnement de travail du logiciel *Flowcode V5 for PICmicros*

2-1 Barre des symboles ou icônes

Pour dessiner un organigramme, Flowcode utilise les symboles normalisés (Figure 2). On les insère dans le programme par un glisser-déposer à partir de la barre des symboles.

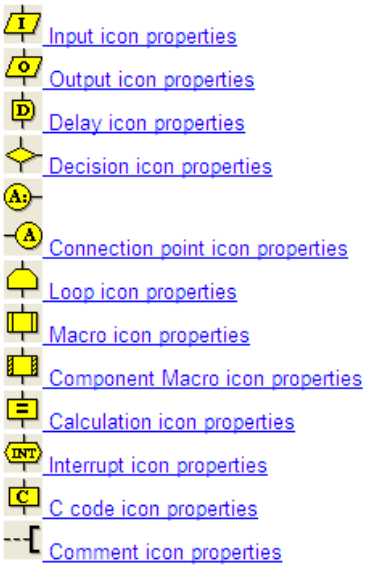


Figure 2 : Barre des symboles ou icônes

2-2 Barre des composants de simulation

Avant de programmer le microcontrôleur PIC, on peut vérifier par simulation le fonctionnement du programme. Les composants peuvent être connectés au PIC pour réaliser cette vérification (Figure 3).

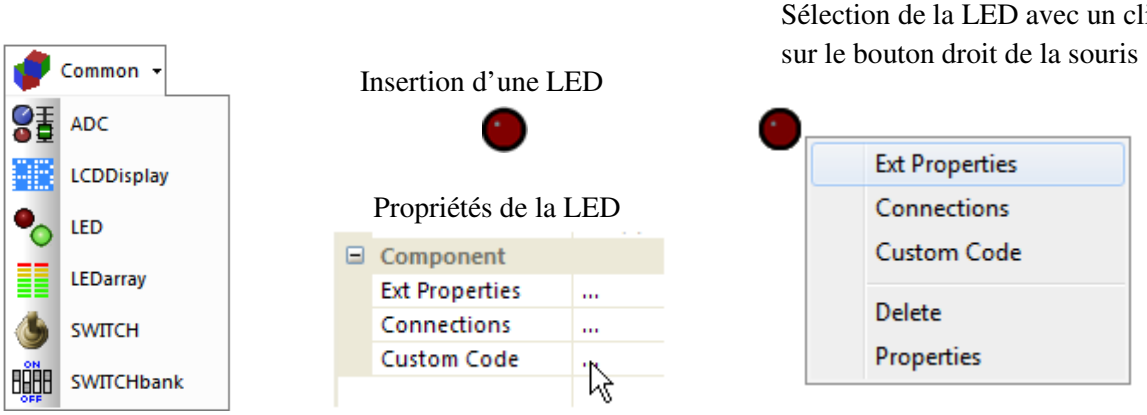


Figure 3 : Exemple d'insertion d'une LED à partir de la barre des composants

2-3 Barre des commandes

Cette barre d'outils propose des boutons pour les commandes de Flowcode les plus couramment utilisées comme Fichier -> Ouvrir. On trouve aussi les boutons démarrer, arrêter et suspendre la simulation (Figure 4).

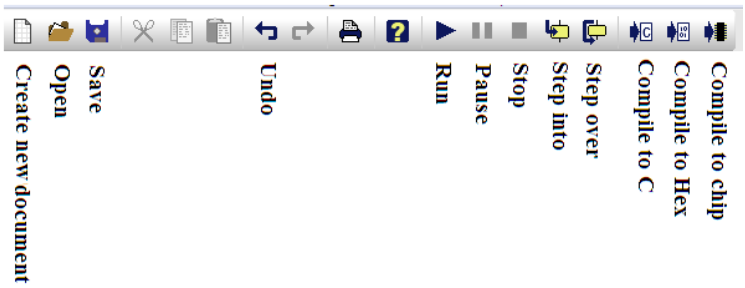


Figure 4 : Exemple des boutons de la barre des commandes

3-Lancement du logiciel Flowcode

Pour l'ouverture ou la création d'un nouveau projet contenant un organigramme représentant l'application développée, il faut sélectionner soit « open » ou « create » (Figure 5).

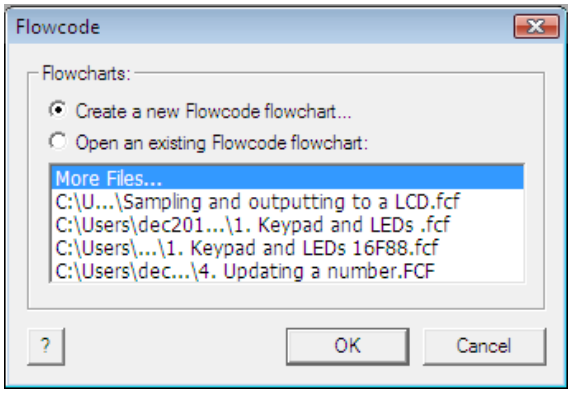


Figure 5 : fenêtre d'ouverture d'un projet

Avant d'utiliser Flowcode pour développer une application, vous devez d'abord choisir un microcontrôleur cible. Le choix de ce dernier s'effectue en activant la fenêtre suivante :

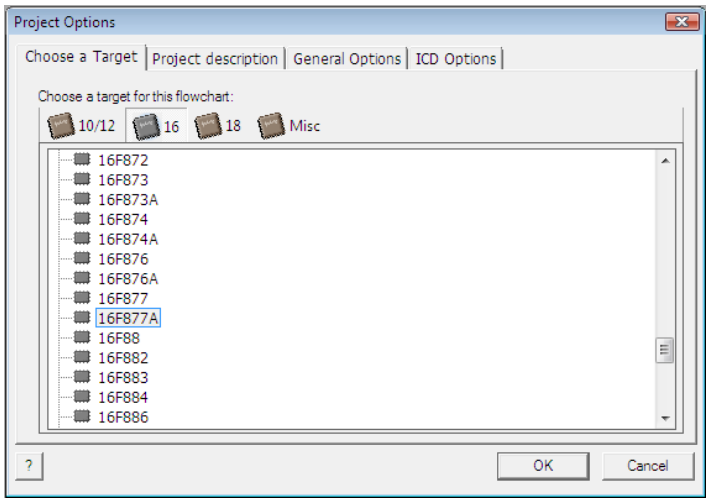


Figure 6 : Sélection du microcontrôleur utilisé

Une fois que vous avez choisi le circuit intégré dont vous avez besoin, vous aurez d'autres choix à faire concernant ce circuit (vitesse d'horloge, entre autres) (Figure 7). Donc, avant d'introduire un programme dans votre microcontrôleur, vous devez tout d'abord suivre ces quelques étapes pour initialiser un certain nombre de données vitales pour le microcontrôleur.

a- Sélection d'un microcontrôleur ;
b- Choix de la vitesse d'horloge ;
c- Configuration des paramètres (vitesse de simulation, fusibles (fuses), ..etc).

Flowcode doit connaitre la vitesse à laquelle est cadencé le microcontrôleur utilisé de façon à pouvoir implémenter la fonction pause (delay) avec les bons paramètres. Vous pouvez modifier la vitesse d'horloge du microcontrôleur en sélectionnant la commande Build -> Project Options, puis entrez la valeur correcte du circuit d'horloge (Figure. 7). Outre le type d'oscillateur, il y a normalement bien d'autres paramètres définissables pour une utilisation des caractéristiques avancées telles que temporisateur (watchdog timer) ou détection de tension d'alimentation faible (Brown Out Detect). La configuration manuelle des fusibles du microcontrôleur se fait en cliquant sur le bouton « Configure chip » après avoir décoché la case «Use external program to set configuration options» de la rubrique Build -> Comiler options-> Programmer (Figure 8). Donc, il faut coller les mots hexadécimaux dans la zone de texte qui s'active avec la fenêtre «Configure» après un clic sur le bouton « Configure chip » (Figure 9). Ces mots hexadécimaux peuvent être obtenus à partir de la de configuration automatique des fusibles du compilateur MikroC (Figure 10).

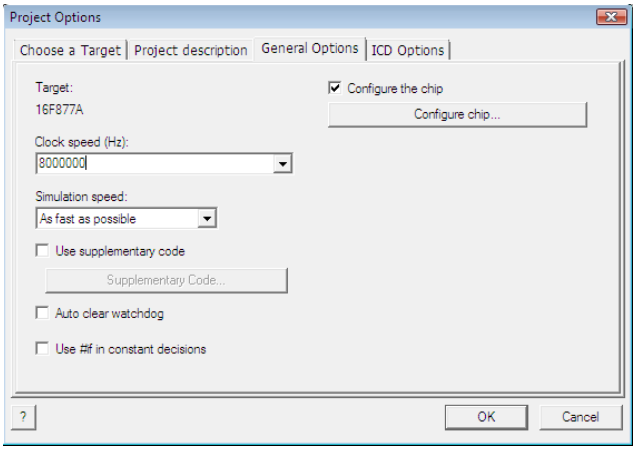


Figure 7 : Configuration des options du projet

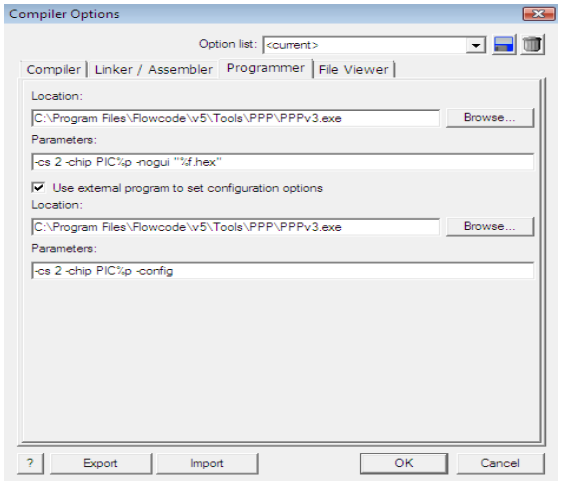


Figure 8 : Configuration des options du projet

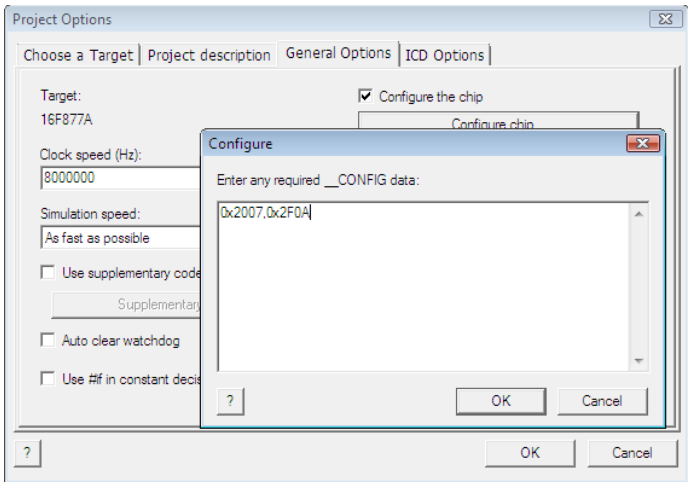


Figure 9 : Les mots hexadécimaux de la configuration des fusibles

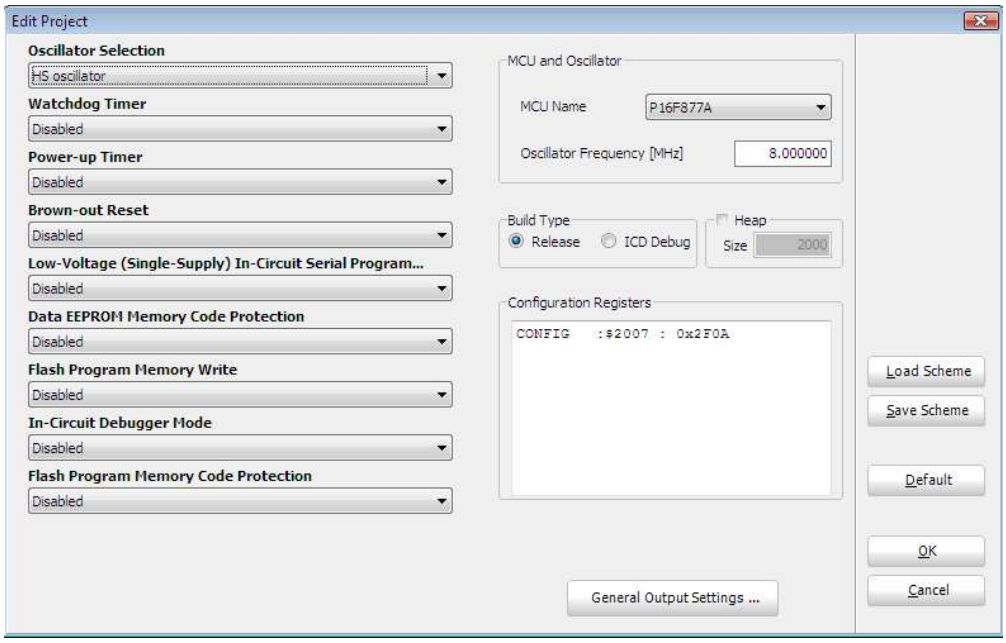


Figure 10: Fenêtre de configuration automatique des fusibles du compilateur MikroC

La deuxième alternative est représentée par la fenêtre suivante qui correspond à la configuration très simple et automatique des fusibles du PIC microcontrôleur avec PICmicro (PICmicro Parallel Programmer (PPP), Matrix Multimedia Ltd) sans la nécessité de passer par un compilateur. Selon la figure 8, la fenêtre de la figure 11 correspondante au logiciel de programmation PICmicro appartenant au logiciel Flowcode s'activera dès que l'on clique sur le bouton « Configure chip ». Une troisième alternative correspondante à la configuration des fusibles peut se faire lors de la phase de chargement du programme dans le microcontrôleur. Donc, l'interface correspondante à cette phase est donnée par la fenêtre du logiciel de programmation de PIC « mikroProg suite for pic [v2.32] » de la figure .12.

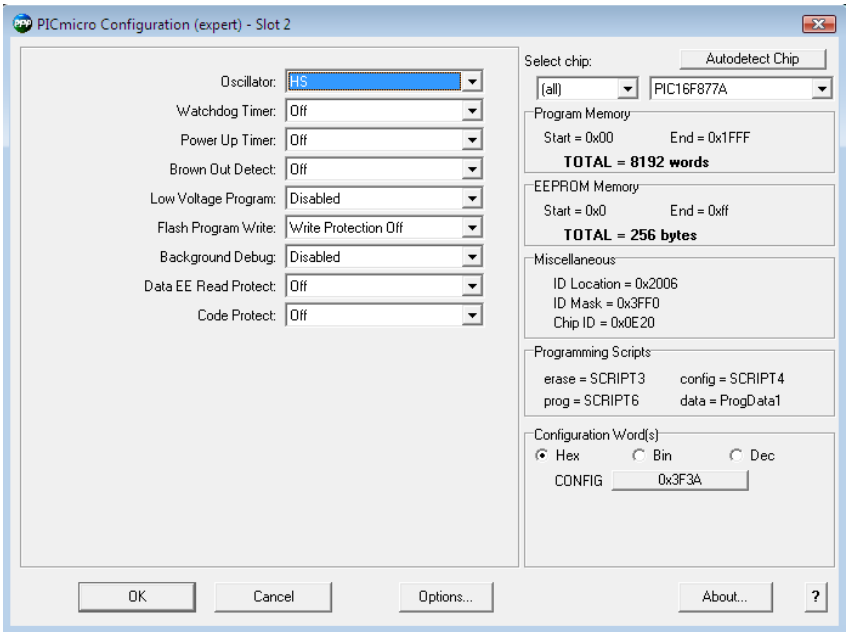


Figure 11 : Configuration automatique des fusibles

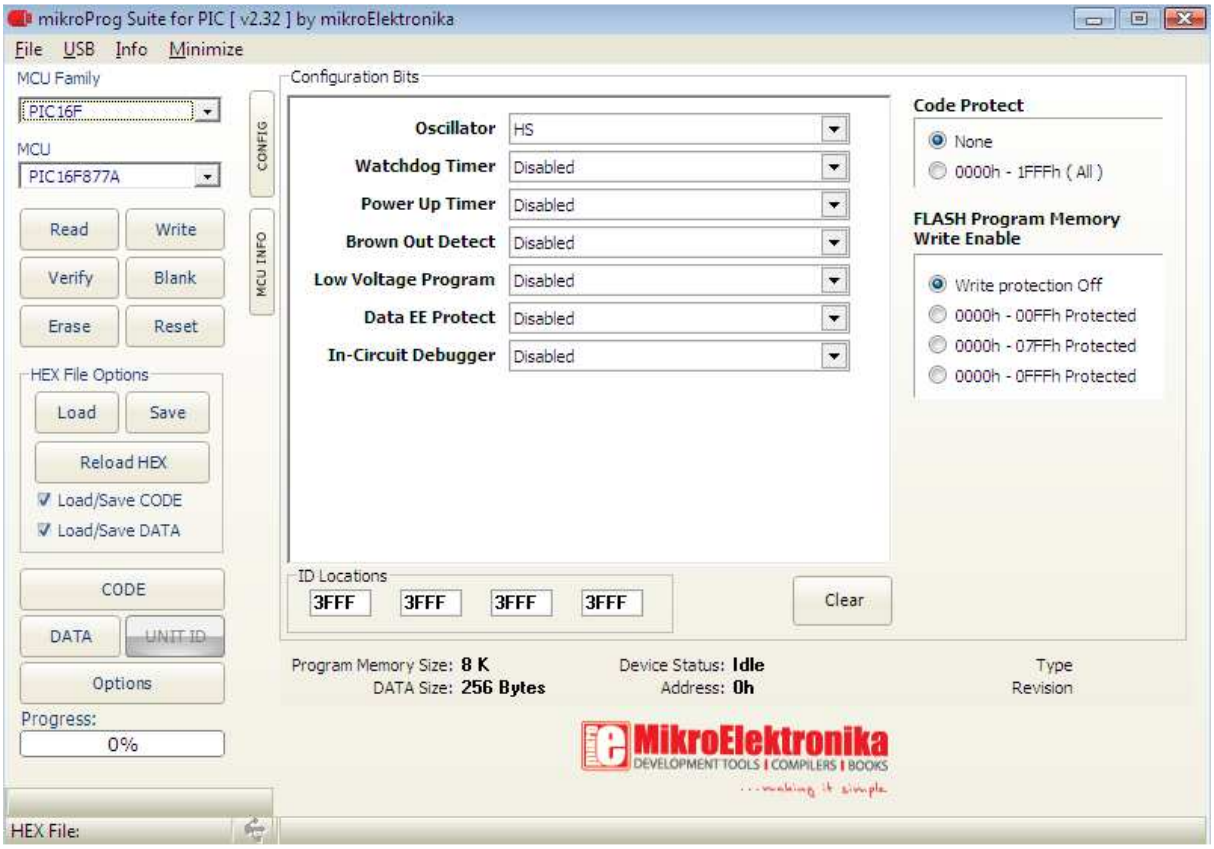


Figure 12 : Configuration automatique des fusibles avec mikroProg Suite

4- Propriétés des entrées/sorties

L'icône d'entrée de la figure 13 lit le port spécifié (ou certains bits seulement du port) et place le résultat dans la variable spécifiée (port A -> BP2, variable BP2 est de type Byte).

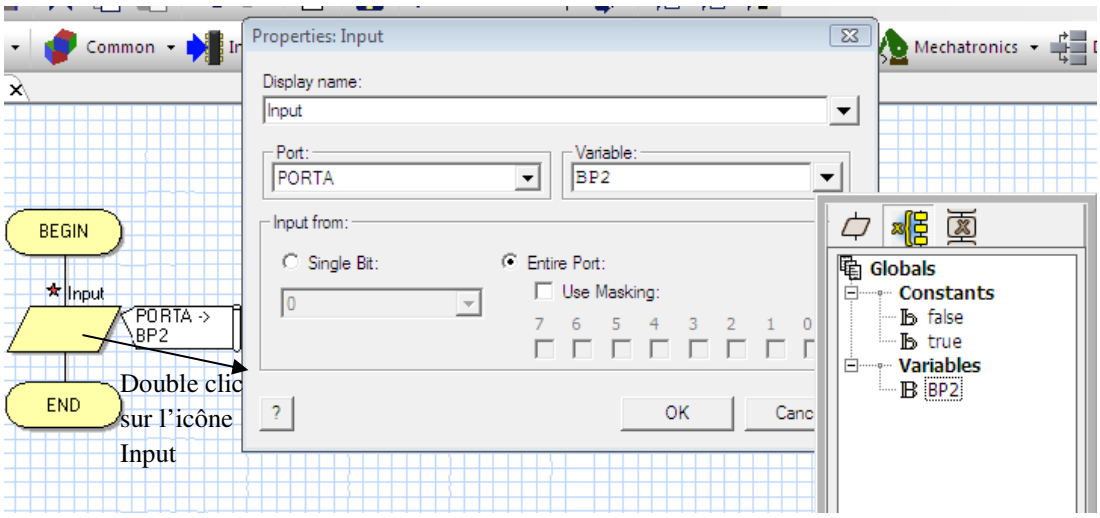


Figure 13 : Opération d'entrée

Nom à afficher ‘Input’: Le texte qui apparaîtra en haut et à droite de l'icône sur l’organigramme.
Variable ‘BP2’: Sélectionner le nom d’une variable dans laquelle vous souhaitez placer le résultat de la lecture des bits du port.
Bouton Variables : Ce bouton ouvre une boite de dialogue permettant de choisir une variable existante ou d’en créer une nouvelle.
Port : Choisir le Port concerné parmi la liste des ports disponibles du microcontrôleur à programmer.
Entrée depuis Bit unique : Utiliser cette option pour lire l’état d’un seul bit du port.
Entrée depuis Port complet : Décocher Masque. Utiliser cette option pour lire l’état du port en entier et ranger la valeur lue dans la variable choisie.

Grâce au masquage, il est possible de transférer seulement certains bits dans une variable. Quand un masque est utilisé, seules les valeurs correspondant aux bits du port sélectionnés sont lues.

L’icône de sortie de la figure 14 permet d’envoyer la valeur ou le contenu d’une variable au port ou aux bits spécifiés du port. La sortie est reçue par le port en format binaire.

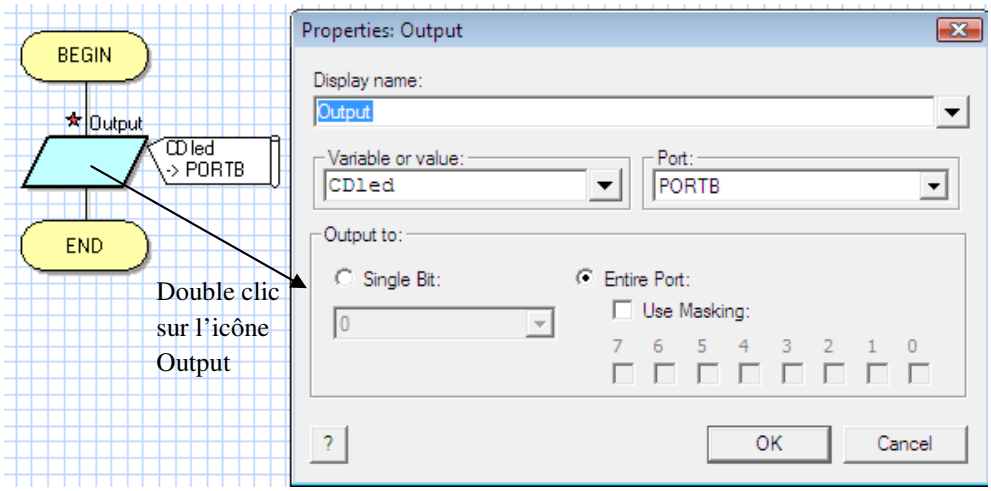


Figure 14 : Opération de sortie

Nom à afficher ‘Output’: Le texte qui apparaîtra en haut et à droite de l'icône sur l’organigramme.
Variable ou valeur ‘CDled’: Sélectionner la variable ou la valeur numérique au format décimal (type par défaut) que vous souhaitez écrire dans ce port.
Port : Le sélectionner depuis la liste des ports disponibles sur le PIC à programmer.
Sortie vers Bit unique : Utiliser cette option pour écrire dans un seul bit du port.
Sortie vers Port complet : Décocher masque. Utiliser cette option pour écrire la valeur ou la variable dans le port entier.

Grâce au masquage, il est possible d'écrire seulement certains bits d'une variable dans un port. Quand un masque est utilisé, seuls les bits sélectionnés sont affectés par l'opération d'écriture.

5- Propriétés de l'icône boucle

Les icônes boucle sont utilisées pour mettre en œuvre des structures itératives (répétitives). Cinq types d'itérations sont réalisables (figure. 15) :

- La boucle "Tant que … Faire…" (test de la boucle au début)
- La boucle "Faire… Tant que …" (test de la boucle à la fin)
- La boucle "Répéter… Jusqu'à …" (test de la boucle au début)
- La boucle "Jusqu'à… Répéter …" (test de la boucle à la fin)
- La répétition de boucle un nombre de fois spécifié (de 1 à 255)

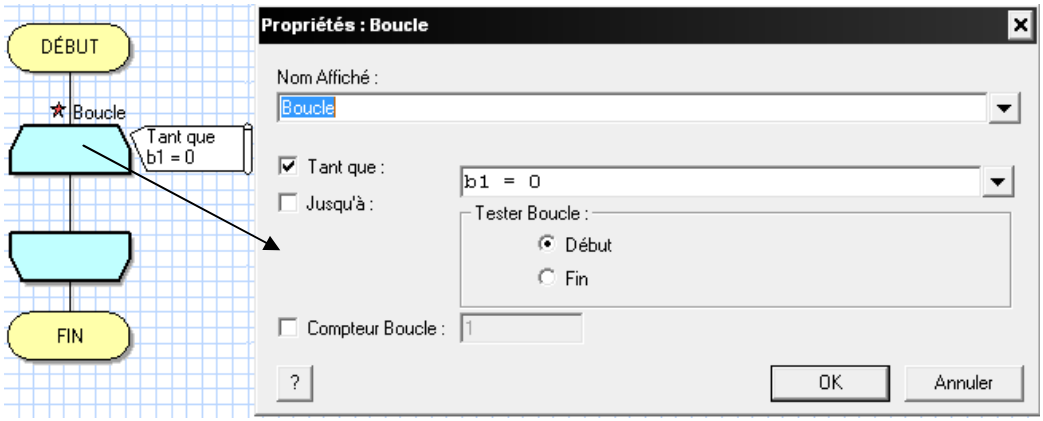


Figure 15 : Boucle itérative tant-que faire

Nom à afficher : Texte à afficher sur l'organigramme en haut et à droite de l'icône.
Tant que, Jusqu'à, Compteur de boucle : Sélectionner le type de structure itérative. Entrer la condition qui permet de rester dans la boucle (boucle "Tant que…") ou de sortir de la boucle (boucle "Répéter…")
Bouton Variables: Ce bouton ouvre la boite de dialogue des variables vous permettant de sélectionner une variable existante ou d'en créer une nouvelle. Ce bouton se trouve à droite de la case condition (b1=0).
Tester la boucle : Cette option permet de spécifier si la boucle doit être testée au début ou à la fin de la boucle.

Il arrive qu'une tâche soit répétée à l'infini (exemple : scrutation). Une façon pratique d'obtenir ce fonctionnement est d'utiliser une boucle infinie. Tester une condition "Toujours Vrai" c.-à-d. "1" fera que la boucle sera répétée indéfiniment.

6- Propriétés des icônes décision et multi-décision

L'icône de Décision, appelée aussi alternative, permet de tester une condition et d'effectuer certains traitements en fonction du résultat du test (Figure 16).

Nom à afficher : Texte à afficher sur l'organigramme en haut et à droite de l'icône.
Si :Le losange Décision teste la condition afin de déterminer dans quelle branche se passera la suite du traitement (Figure 17). Si le résultat du test vaut 0 ou FAUX, c'est la branche 'Non' qui sera déroulée. Si le résultat du test vaut un nombre différent de 0 ou VRAI alors c'est la branche du "Oui" qui sera exécutée. Les tests peuvent contenir des nombres, des variables et des opérateurs.
Bouton Variables : Ce bouton ouvre la boite de dialogue Variables permettant de sélectionner une variable existante ou d'en créer une nouvelle.

Inverser Oui et Non : Normalement la branche correspondant à "Oui" part sur la droite de l'icône de décision et la branche correspondant au 'Non' continue tout droit dans l'organigramme. Cocher cette option pour inverser les deux branches.

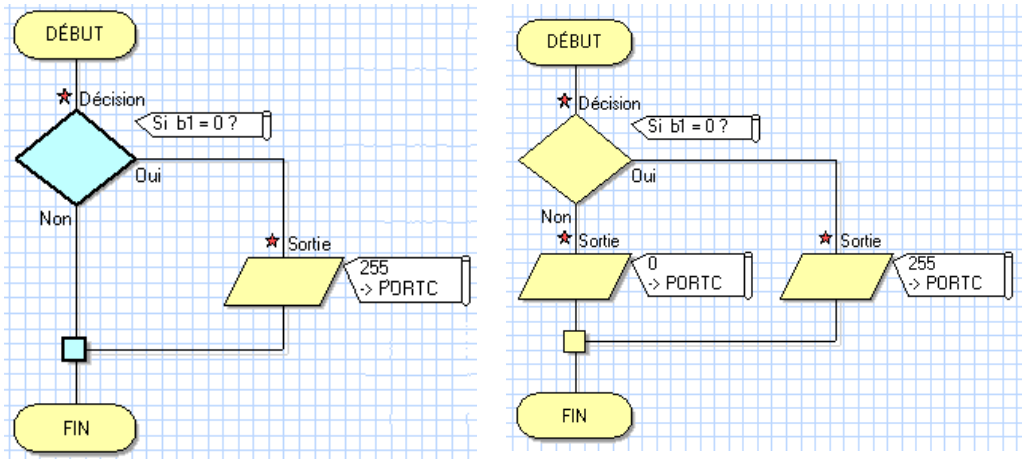


Figure 16: Alternative simple et composée

Figure 17: Propriétés de l'alternative

La figure 18 représente une structure multi-décision. Si la variable b1=1 le traitement 1->C1 s'effectue, si b1=2 le traitement 1->C2 s'effectue et dans tous les autres cas le traitement par défaut c.-à-d. 1->C0 s'effectue.

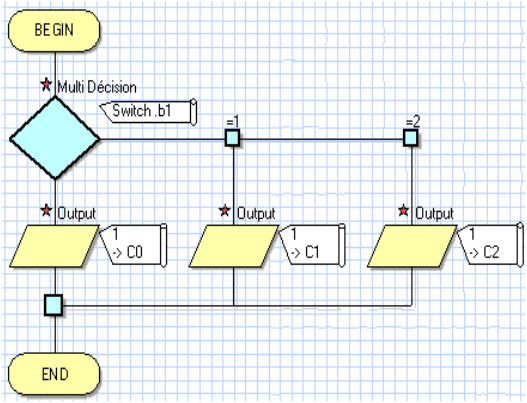


Figure 18: Exemple d'une alternative multiple

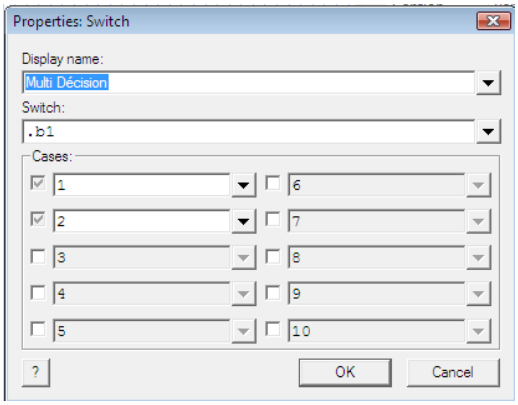


Figure 19: Propriétés de l'alternative multiple

7- Propriétés de l'icône calcul

L'icône de Calcul permet la modification des variables. Elle peut être utilisée pour vérifier des entrées ou créer des sorties (Figure 20).

Nom à afficher : Texte à afficher sur l'organigramme en haut et à droite de l'icône.
Calculs : Une ou plusieurs lignes de calculs peuvent être entrées dans cette boite de dialogue.
Tous les calculs doivent comprendre le nom d'une variable existante, le signe égal suivi d'une expression faite de nombre, de variables et des opérateurs suivants :
( ) : Parenthèses.
=, <> : Egal à, Non égal à.
+,-,*, / ,MOD : Addition, Soustraction, Multiplication, Division, Modulo (reste de la division entière).
< , <=, >, >= : Plus petit que, plus petit ou égal à, Plus grand que, Plus grand ou égal à.
>> ,<< : Décalage à droite, décalage à gauche.
NOT AND (&) OR (|) XOR (^) : NON (inversion), ET, OU, OU Exclusif (opérations bit à bit) ! && || - NON, ET, OU (opérations sur octet(s), le résultat vaut 0 ou 1)

Bouton Variables : Ce bouton ouvre la boite de dialogue des variables afin de choisir une variable existante ou d'en créer une nouvelle.

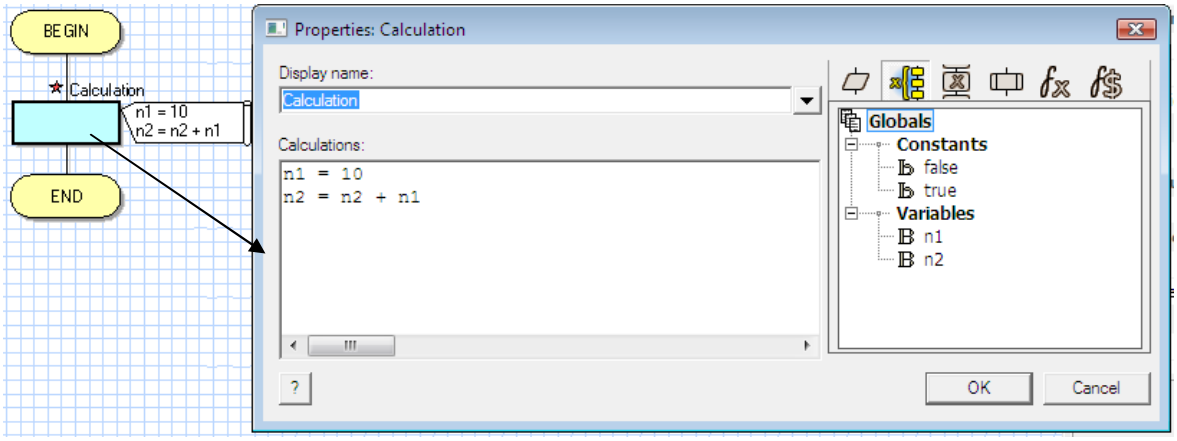


Figure 20 : Propriétés de l'icône calcul

8- Propriétés des icônes code C et commentaire

Des programmes écrits en C (ou/et en Assembleur) peuvent être ajoutés à une application Flowcode grâce à l'icône Code C (Figure 21). Ce code ne pourra pas être simulé par Flowcode, mais sera transmis au microcontrôleur durant la compilation.

Nom à afficher : Le texte qui apparaîtra en haut et à droite de l'icône sur l'organigramme.
Code C : Entrer le code C que vous souhaitez inclure à votre organigramme. Le code C n'est pas contrôlé par Flowcode mais est transmis directement au compilateur C lorsque l'organigramme est compilé. Il est important de vérifier que le code C entré est correct, puisque les erreurs éventuelles de syntaxes feront échouer la compilation de tout votre organigramme. Pour accéder aux variables Flowcode, aux macros et aux points de jonction, il est nécessaire de caractériser l'élément dans votre code C par les préfixes respectifs FCV_, FCM_ et FCC_NomMacro_. Par exemple, pour utiliser la variable Flowcode appelée TEMPO dans votre code C, vous devrez y faire référence en utilisant FCV_TEMPO. Notez que toutes les variables définies avec Flowcode sont écrites en majuscules.

Pour utiliser la macro Flowcode appelée TEST dans votre programme en C, vous devrez l'appeler FCM_TEST(). Notez que tous les noms de macros Flowcode doivent s'écrire en majuscules.

Pour aller à un point de jonction nommé A, défini dans une macro Flowcode nommée TEST, votre code C doit y faire référence par FCC_TEST_A.. Les points de jonction définis dans l'organigramme principal de Flowcode doivent contenir le préfixe FCC_Main_.

Il est possible d'entrer des instructions assembleur dans la fenêtre de Propriétés du code C. Pour une ligne d'assembleur, utiliser l'opérateur **asm** devant l'instruction, par exemple :
asm movlw 5
Vous pouvez aussi spécifier plusieurs lignes d'assembleur. Procédez de la façon suivante pour encadrer plusieurs instructions à l'intérieur d'un bloc asm:

```
asm
{
        ; Entrer votre code ici
}
```

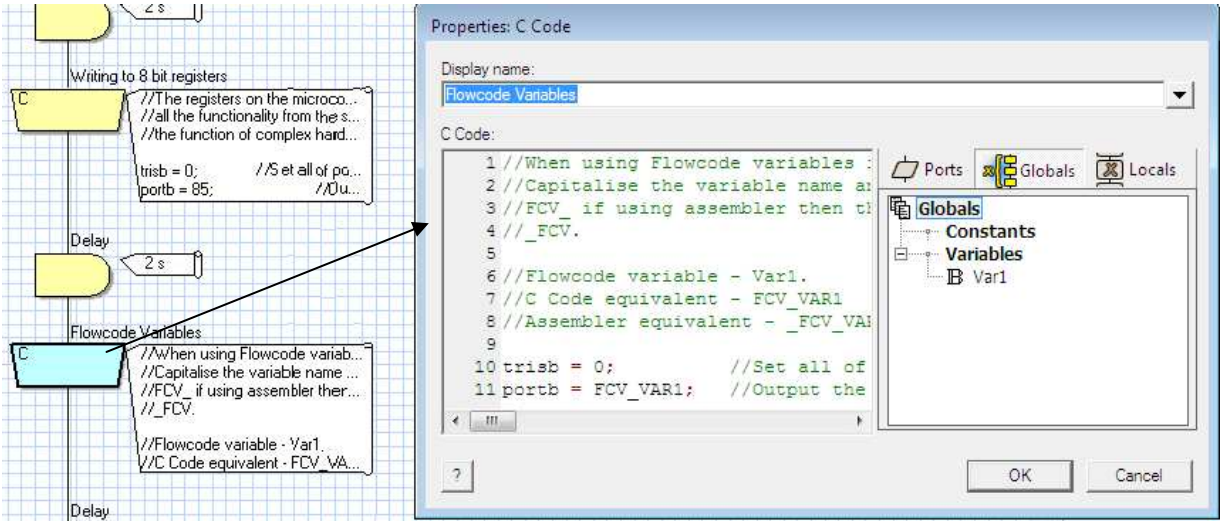


Figure 21 : Insertion d'instructions en code C

Les commentaires vous permettent d'annoter votre code pour en expliquer les fonctions, vous souvenir de l'utilité d'une variable, de certains paramètres, etc. L'icône Commentaires vous permet de placer des commentaires sur votre organigramme. Tirez l'icône et déposez-la à l'endroit du code où vous voulez introduire un commentaire puis servez-vous de la fenêtre des propriétés des icônes.
Un commentaire ne doit pas être exécuté et ne fait donc pas partie du programme en tant que tel.

9- Propriétés de l'icône macro

Les macros sont des portions de code réutilisables dans un projet (Figure 22). Les macros dans Flowcode sont divisées en deux catégories : les macros logiciellles et les routines composant. Les routines composant sont des macros prédéfinies qui accompagnent les composants fournis par Flowcode. Par exemple, les macros LCD permettent d'afficher des

caractères alphanumériques sur l'écran LCD. Une routine Composant fonctionne uniquement avec un composant déterminé. L'icône de la routine Composant se reconnaît aux bandes hachurées sur le bord extérieur.

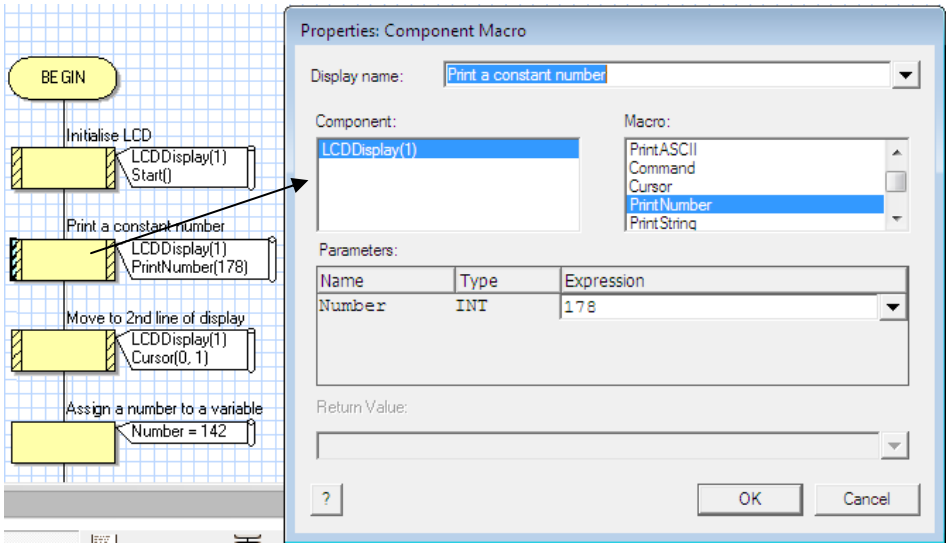


Figure 22 : Exemple d'utilisation du composant LCD

L'utilisateur peut concevoir et écrire ses propres macros et la bordure extérieure de l'icône est claire et sans hachures. Un double clic sur l'icône ouvre le menu de Propriétés des macros et permet à l'utilisateur de sélectionner ou d'ajouter des macros (Figure 23). Cliquez sur Nouvelle Macro pour démarrer l'écriture d'une nouvelle macro à ajouter à la liste. Saisissez alors tous les paramètres requis et sélectionnez une valeur de retour si nécessaire.

Si la macro a besoin de paramètres, ils doivent être introduits dans ce champ. Il peut s'agir de valeurs numériques ou de variables existantes. Chaque variable ou valeur doit être séparée par une virgule dans la liste. Le détail des paramètres affichera le type de chaque paramètre. Pour être acceptés, les paramètres doivent être du type déclaré. Le type de variable pour la valeur de retour sera affiché. En effet, il faut utiliser une variable du type adéquat pour accueillir la valeur de retour.

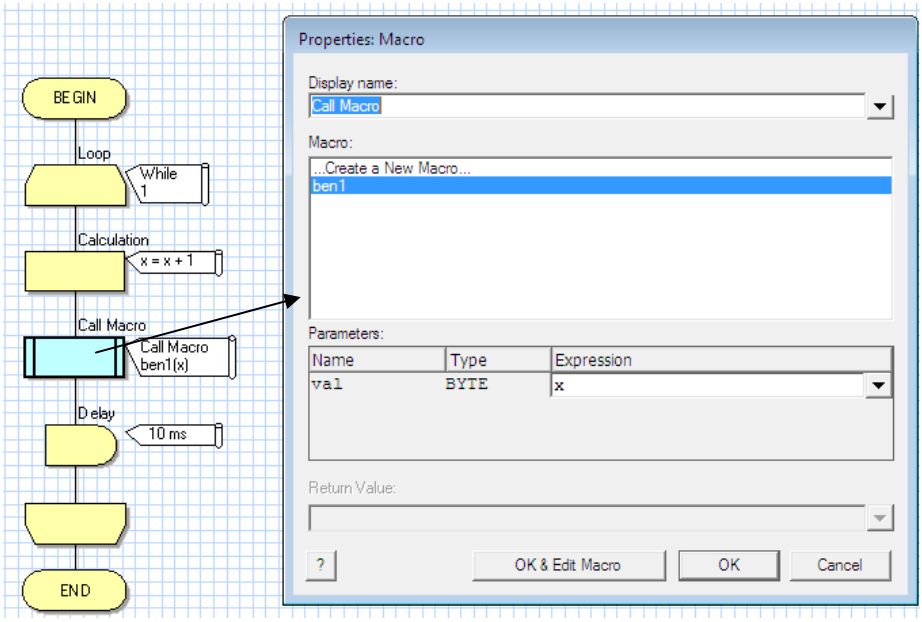


Figure 23 : Exemple d'utilisation de la macro logicielle

10- Propriétés de l'icône interruption

Les interruptions servent à réagir à des événements tels qu'un stimulus externe ou l'intervention d'une horloge interne. Le nombre et le type d'interruptions disponibles dépendent du microcontrôleur utilisé. Les caractéristiques et le mode opératoire varient d'une interruption à l'autre. L'utilisateur devra se référer aux boites de dialogue pour obtenir les précisions (Figure 24). Cependant Flowcode se sert de quatre type principaux :

TMR<X> - overflow : réagit à une fin de temporisation liée à un "timer" interne du processeur.
INT : réagit à un changement d'état logique sur une broche du processeur configurée en interruption externe.
Port change : réagit à un changement d'état logique sur un ensemble de broches du processeur.
Défini par l'utilisateur : c'est l'utilisateur qui détermine la procédure d'interruption.

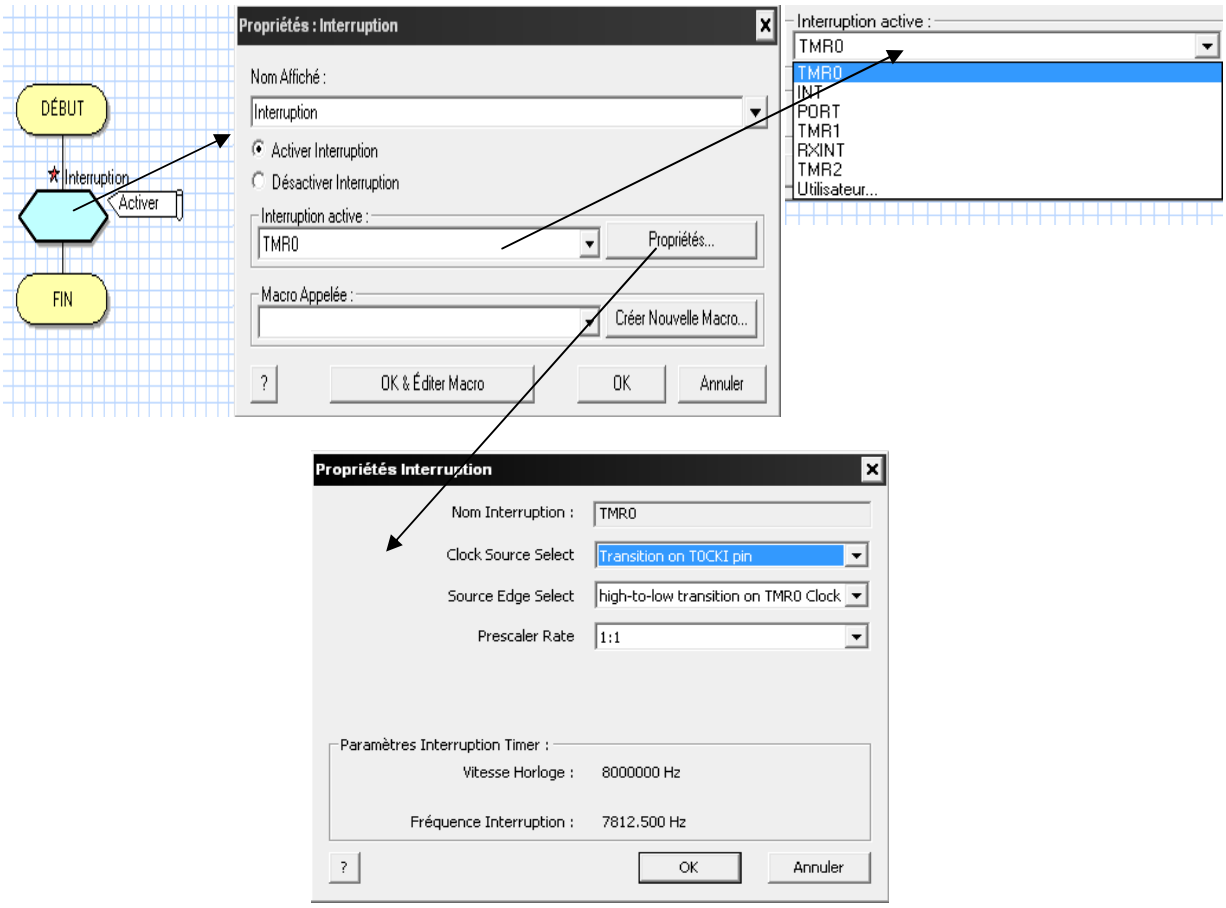


Figure 24 : Exemple de configuration d'une interruption

11- Propriétés de l'icône point de jonction

Les icônes de jonction sont utilisées pour "sauter" d'un point de l'organigramme à un autre (Figure 25). Quand l'organigramme atteint le point de jonction, il saute directement au point de jonction correspondant et continue ensuite l'exécution à partir de ce point. Les icônes de jonction sont utilisées par paires.

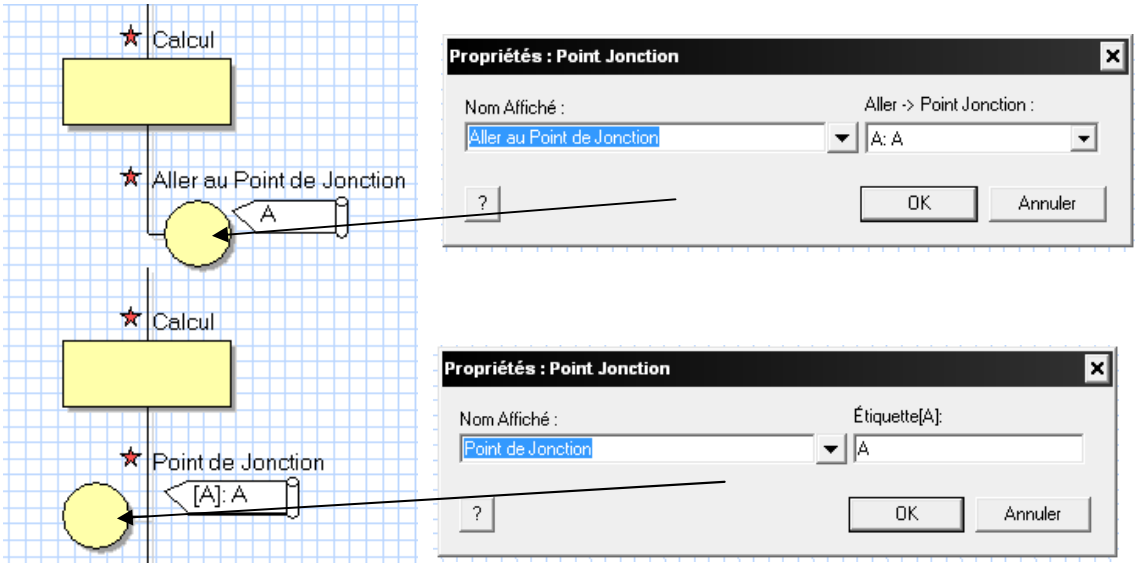


Figure 25 : Point de jonction

Le saut inconditionnel "déstructure" l'organigramme et peut provoquer des dysfonctionnements du système, il ne faut donc l'employer qu'après avoir constaté l'impossibilité d'utiliser une autre solution.

12- Propriétés de l'icône pause

L'icône Pause permet d'insérer des temporisations dans votre programme et d'en ralentir l'exécution (Figure 26). Pendant l'exécution de ces pauses le processeur est entièrement occupé et il ne peut donc effectuer d'autres opérations en mode normal, seul le mode "Interruption" peut lui faire exécuter des opérations.

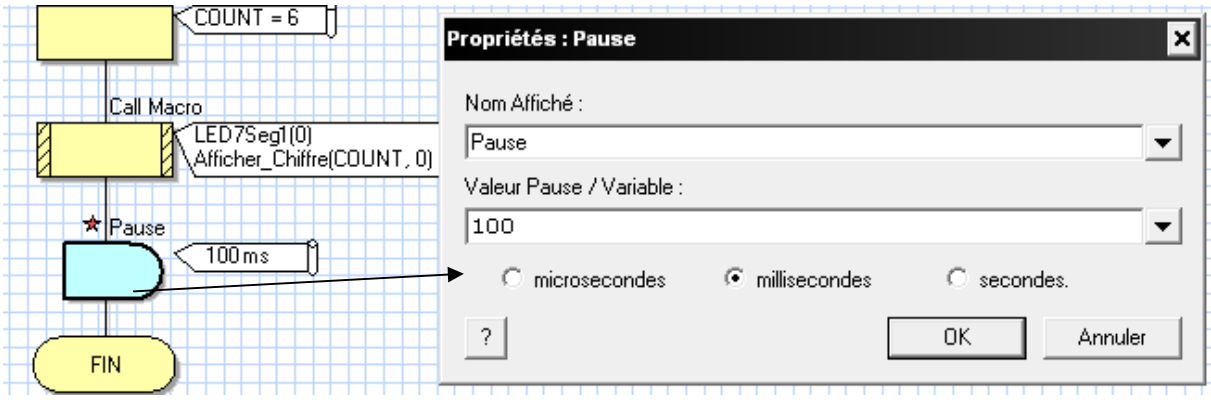


Figure 26 : Propriétés de l'icône pause

13- Exemples d'applications

Exemple N°1: clignotement d'une LED.

Le but de cet exemple est de faire clignoter une LED de couleur rouge connectée à la broche RB0 du port B du PIC16f877A de 8MhZ (Figure 27). La LED s'allume pendant une seconde et s'éteint pendant la seconde suivante. Le programme correspondant est obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika. La figure 28 représente l'organigramme et son programme C développés avec Flowcode.

```
//Programme source
//Editeur du compilateur MikroC Pro V6
void main()
{
 trisb =0x00;//configuration du port B en sortie
   do  // la boucle do-while
    { portb=0x00;//Port B reçoit la donnée 0x00
      delay_ms(1000);// pause d'une seconde
      portb=0x01;//Port B reçoit la donnée 0x01
      delay_ms(1000); //pause d'une seconde
    }while(1);//Boucle infinie
}
```

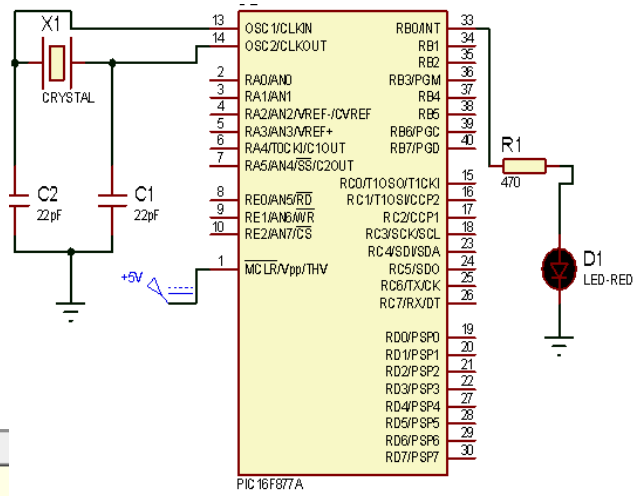
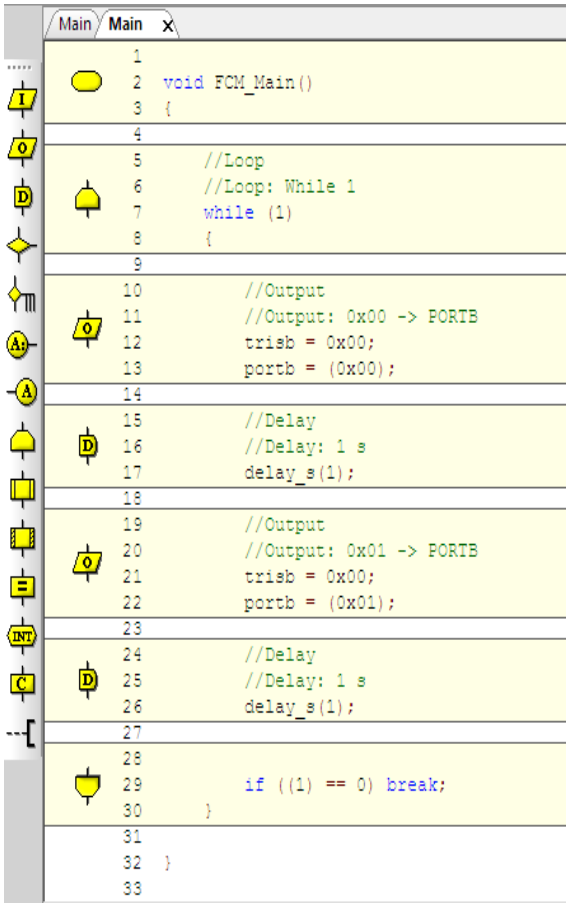


Figure 27:Schéma de connexion de la LED au PIC



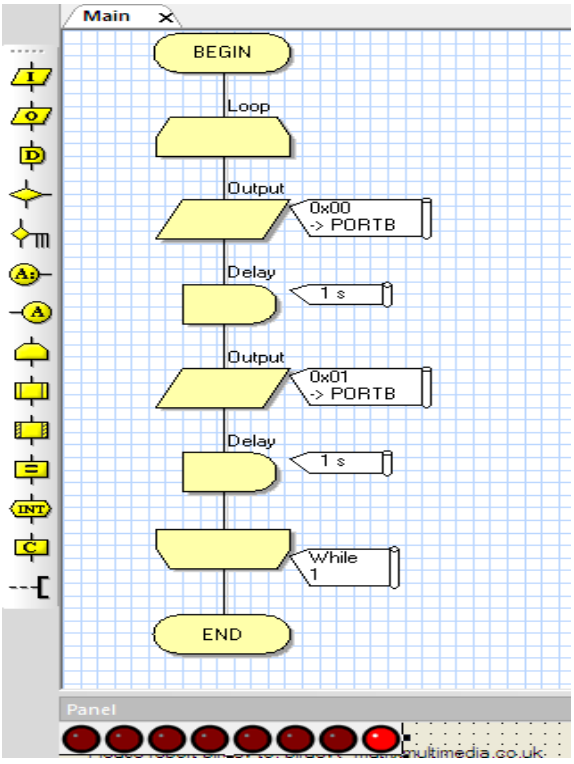


Figure.28: Organigramme et listing du programme source générés par Flowcode

Exemple N°2: clignotement de 8 LED.

Le but de cet exemple est de faire clignoter 8 LED de couleur rouge connectées au port B du PIC16f877A de 8MhZ correspondant aux broches allant de RB0 jusqu'à RB7 (Figure 29). Les LED s'allument pendant une seconde et s'éteignent pendant la seconde suivante.

```
//Programme source
//Editeur du compilateur MikroC Pro V6

void main()
{
  trisb =0x00;
    do
     { portb=0x00;//port B reçoit 0x00
       delay_ms(1000);//pause d'une seconde
       portb=0xff;//port B reçoit 0xFF
       delay_ms(1000);//pause d'une seconde
     }while(1); //boucle do-while (boucle
                    // infinie  avec while(1) )
}
```

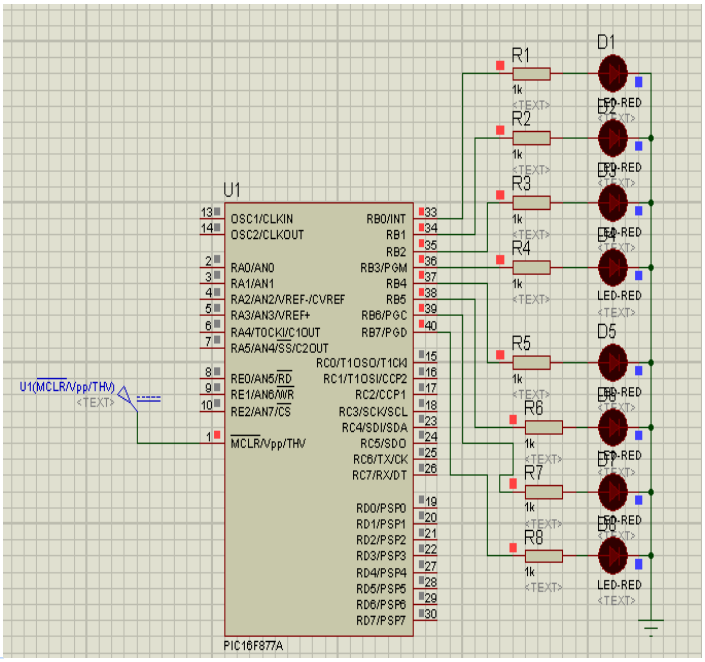


Figure 29:Schéma de connexion des 8 LED au PIC

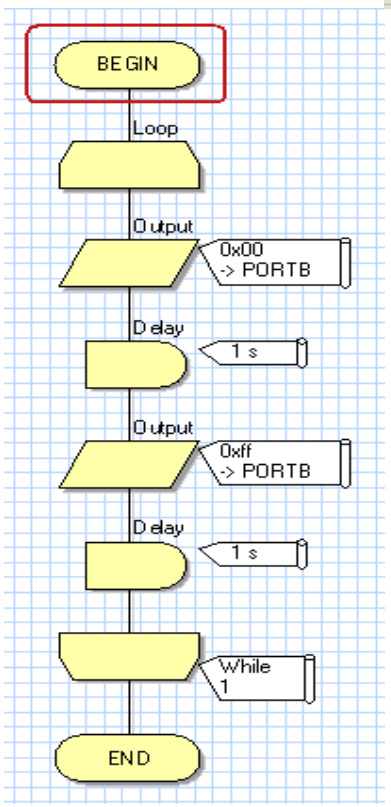


Figure 30: Organigramme correspondant au clignotement de 8 LED

Exemple N°3: Manipulation d'une variable.

Le but de cet exemple est de manipuler une variable de type Byte dont la valeur varie de 0 jusqu'à 255. La même valeur sera transmise au port B du PIC16f877A de 8MhZ. Les LED rouges sont connectées au port B s'allument en fonction de la valeur de la variable My_OUTPUT (Figure 32). Le programme correspondant obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika est le suivant :

```
//Programme source
//Editeur du compilateur MikroC Pro V6
void main()
{
    unsigned short My_OUTPUT=11;
    trisb =0x00;
    portb=0x00;
    portb=My_OUTPUT;
}
```

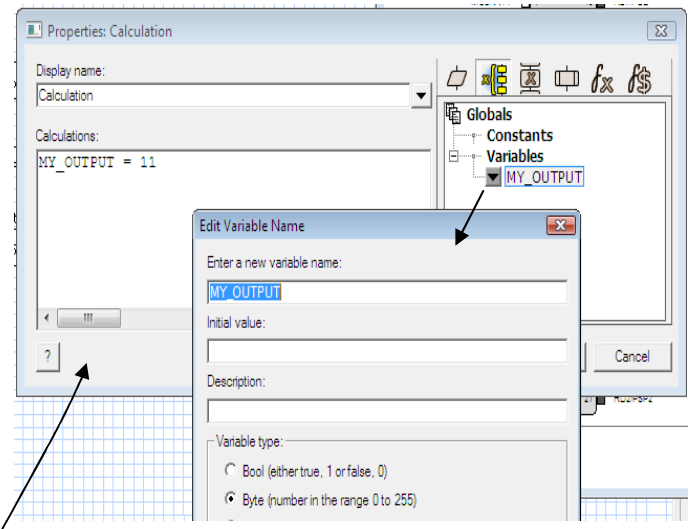


Figure 31: Fenêtre d'ajout d'une variable

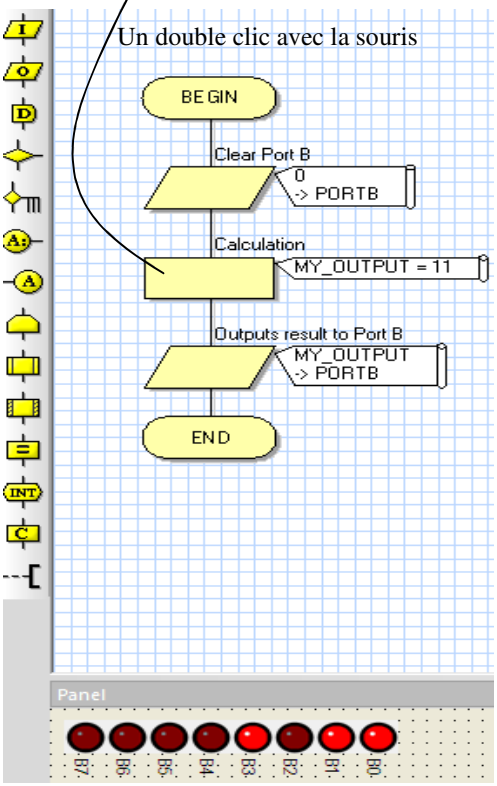


Figure 32: Organigramme correspondant à la manipulation d'une variable

Exemple N°4: La boucle Tant que

Le but de cet exemple est d'utiliser la boucle tant que « while » afin de réaliser un compteur de 8 bits. La variable count est de type byte dont la valeur est comprise entre 0 et 255. Initialement la valeur de count est égale à 0 et sa valeur sera incrémentée à l'intérieur de la boucle infinie while (1) (Figure 34). Donc, à chaque itération la valeur de la variable count est transmise au port B du PIC16f877A de 8Mhz permettant de réaliser un compteur. Le programme correspondant obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika est le suivant :

```
//Programme source
//Editeur du compilateur MikroC Pro V6
void main()

{  unsigned short count;
   trisb=0;
   portb=0;
   count=0;
   while(1)
   { count = count + 1;
     portb=count;
     delay_ms(200);
      }
}
```

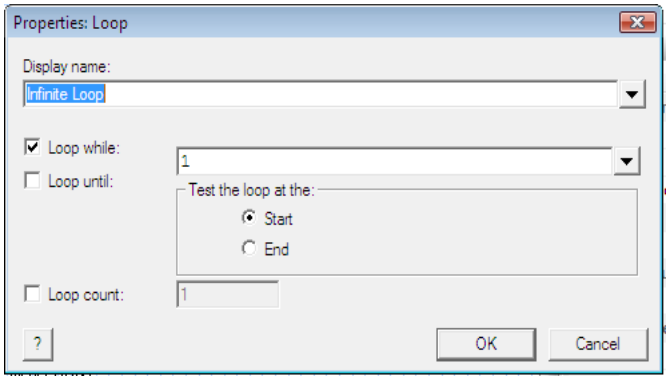


Figure 33: Configuration d'une boucle while

L'organigramme ci-dessus permet de réaliser la même fonction que celle du programme qui compte de 0 jusqu'à 255. Le programme source écrit en langage C, généré par Flowcode est représenté par la figure 31.

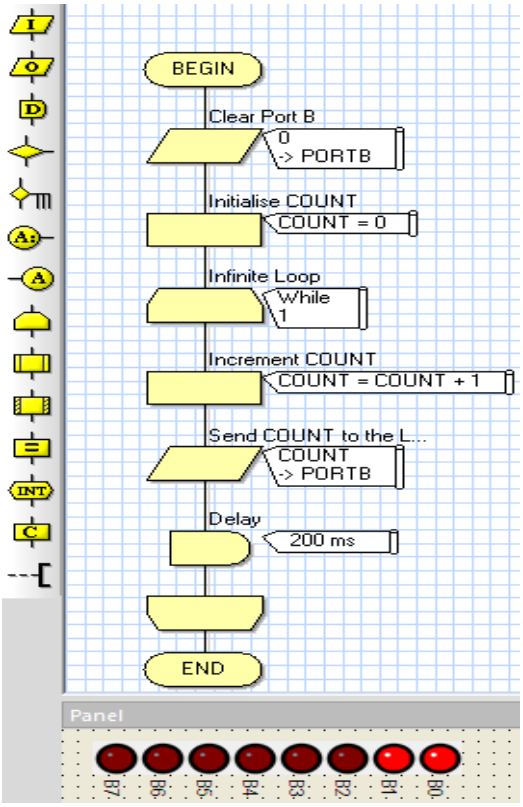


Figure 34: Organigramme correspondant à la manipulation de la boucle while

Le exemple précédent est repris afin d'ajouter une condition à la boucle Tant que (Figure 36). La valeur de la variable count varie de 0 jusqu'à 15 ou à chaque itération cette dernière est incrémentée avec comme condition d'arrêt (count < 15). Le programme correspondant obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika est le suivant :

```
//Programme source
//Editeur du compilateur MikroC Pro V6
void main()

{  unsigned short count;
   trisa=0x00;
   porta=0x00;
   count=0;
   while(count<15)
   { count = count + 1;
     porta=count;
     delay_ms(200);
     }
}
```

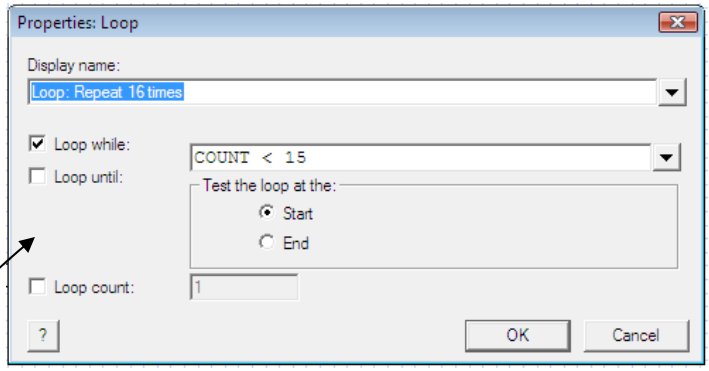


Figure 35: Ajout d'une condition d'arrêt à la boucle while

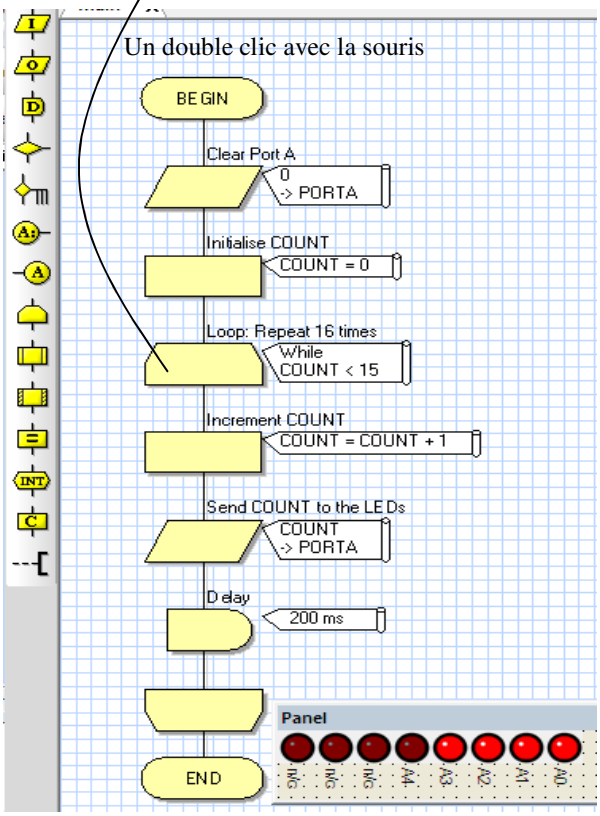


Figure 36: Organigramme correspondant à la manipulation d'une boucle avec une condition d'arrêt

Exemple N°5 : Opération de test « Si-Alors ».

La structure de test simple « Si-Alors » est utilisée dans cet exemple dont le but de réaliser un registre à décalage en exploitant le port B du PIC16f877A de 8MhZ (Figure 37). Les LED rouges sont connectées au port B et s'allument en fonction de la variable count dont la valeur est directement transmise au port B à chaque itération de la boucle infinie « while (1)». Initialement la valeur de la variable count est égale à 1, puis elle est multipliée par 2 afin de réaliser un décalage. Le programme correspondant obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika est le suivant :

```
//Programme source
//Editeur du compilateur MikroC Pro V6
void main() {
unsigned short count;
trisb = 0;
portb = 0;
count=1;
while(1)
{ if(count = = 0)    count=1;
  portb=count;
  delay_ms(200);
  count=count*2;
}
}
```

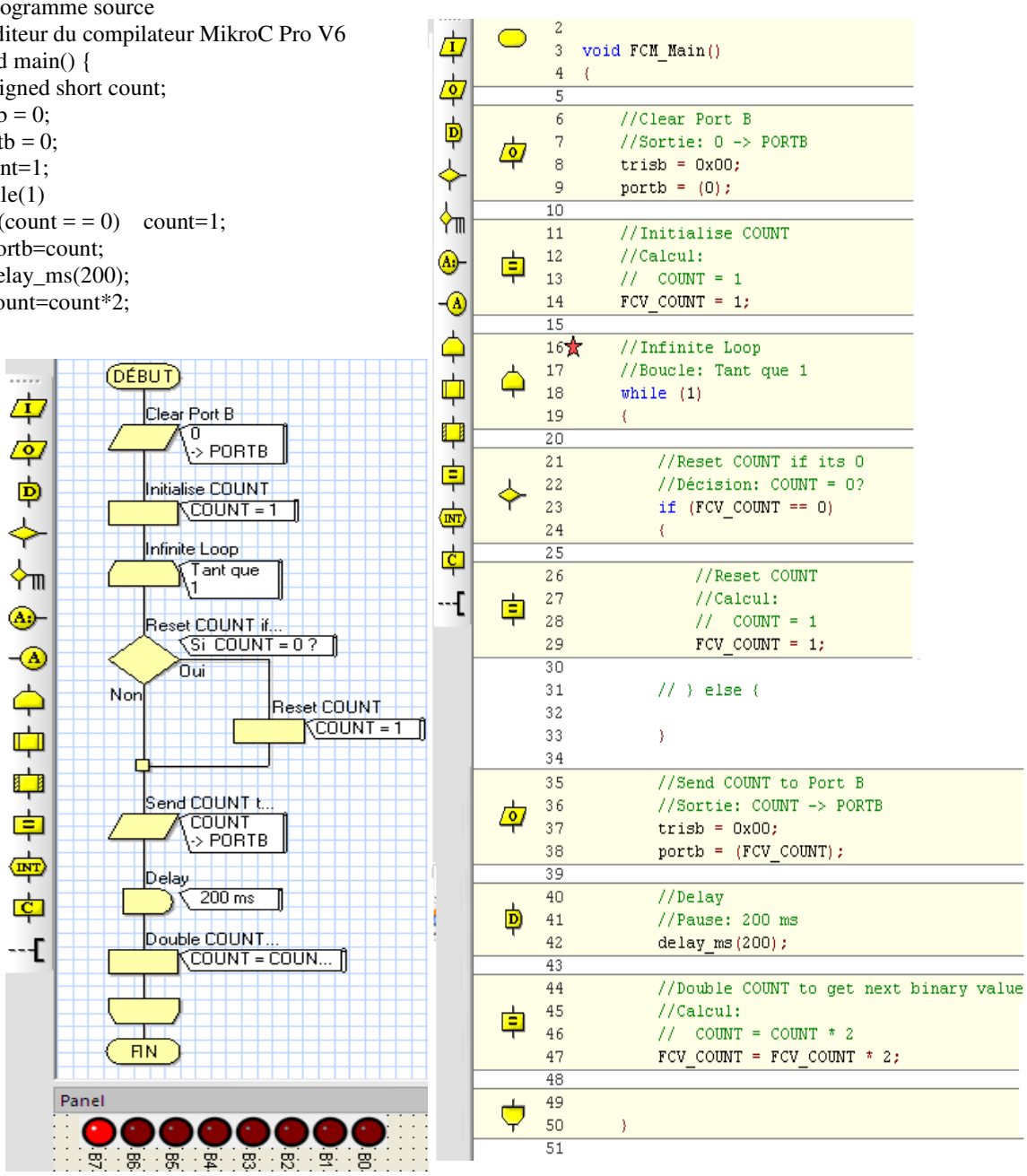


Figure 37: Utilisation d'une structure de test simple dans organigramme

Exemple N°6 : Opération d'entrée.

Le but de cet exemple est de faire une opération d'entrée en lisant la valeur de la broche RB0 du port B par scrutation. Le bouton poussoir est raccordé à la broche RB0 du port B du PIC16f877A de 8MhZ. Le port C est raccordé à 8 LED qui s'allument pendant 200ms dès que le niveau tension change du haut (5V) vers le bas (0V) de la broche RB0 associée au bouton poussoir (Figure 38). Le programme correspondant obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika est le suivant :

```
//Programme source
//Editeur du compilateur MikroC Pro V6
void main()
{  trisb=0xff;
   trisc=0x00;
 do
   {   if(portb.b0 = =0)
       { portc=0xff;
         delay_ms(200);
       }
       portc=0x00;
   }while(1);
}
```

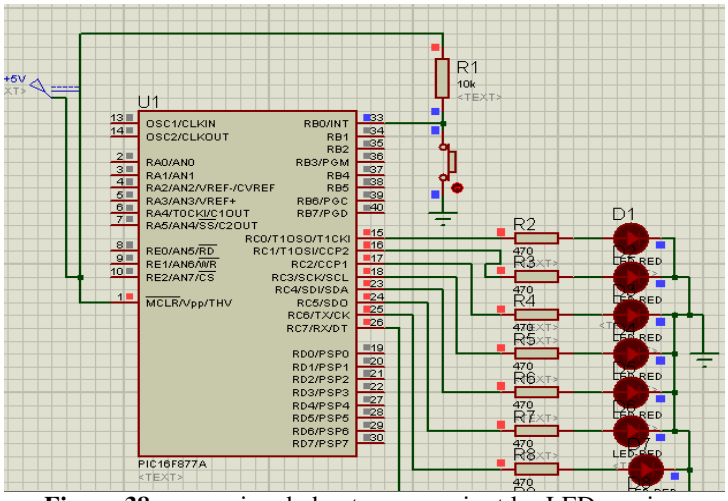


Figure 38: connexion du bouton poussoir et les LED au pic

Concernant l'organigramme correspondant au programme précédent, une variable de type booléenne à été utilisée (port_b0) afin de tester le niveau bas de la broche B0. Un switch est utilisé à la place d'un bouton poussoir selon la figure 39.

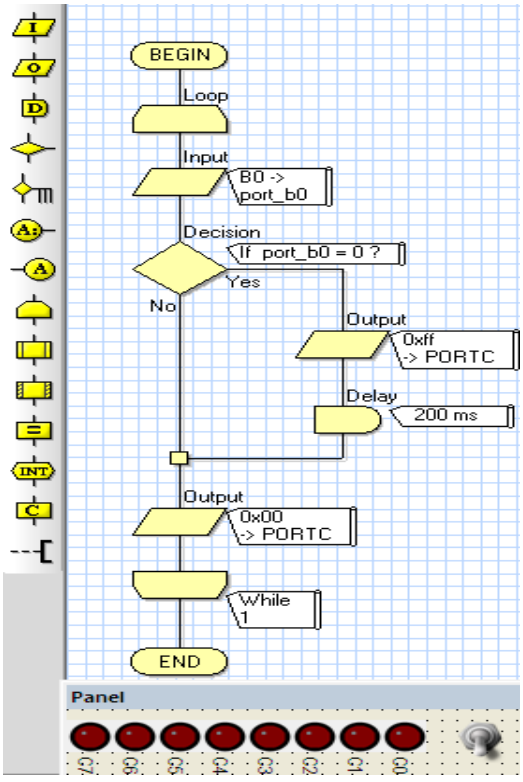


Figure 39: Organigramme utilisant la structure Si-Alors

Exemple N°7 : Interruption.

Cet exemple présente le programme source d'une interruption externe s'effectuant au niveau de la broche RB0 du port B du PIC16f877A de 8MhZ. Le programme principal exécute l'instruction 'portc=count' à travers une boucle infinie se qui permet de fixer les sorties du port C en fonction de la valeur de count. Si une interruption se produit par un changement de niveau de la broche RB0 alors le programme d'interruption va s'exécuter en incrémentant la valeur de la variable count. Ainsi, la valeur de cette variable sera transmise au port C permettant d'allumer les LED correspondantes.

```
unsigned short count ;
void main()
{trisb=0x01;
  trisc=0x00;
OPTION_REG.INTEDG=0;//voir datasheet page 23
INTCON.GIE=1;//voir datasheet page 24
INTCON.INTE=1; //voir datasheet page 24
count=2;
 while(1)
    { portc=count;
    }
}
void interrupt ()
{count = count+1;
INTCON.INTF=0;
 }
```

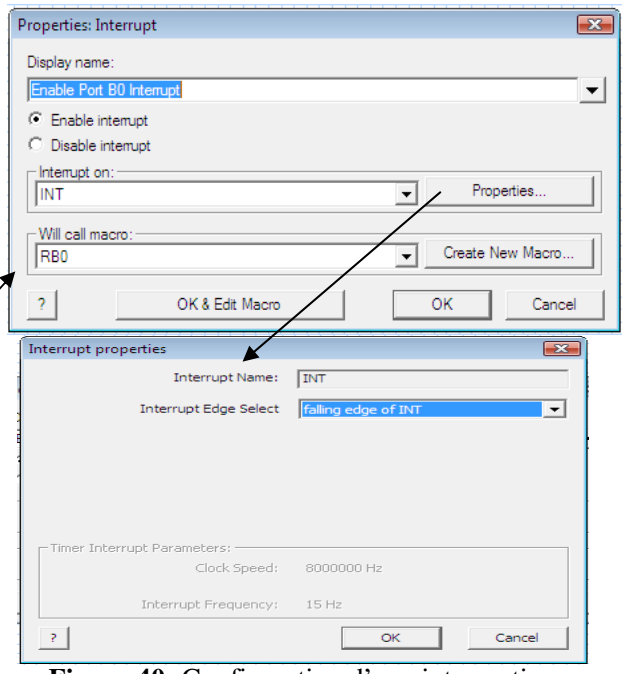
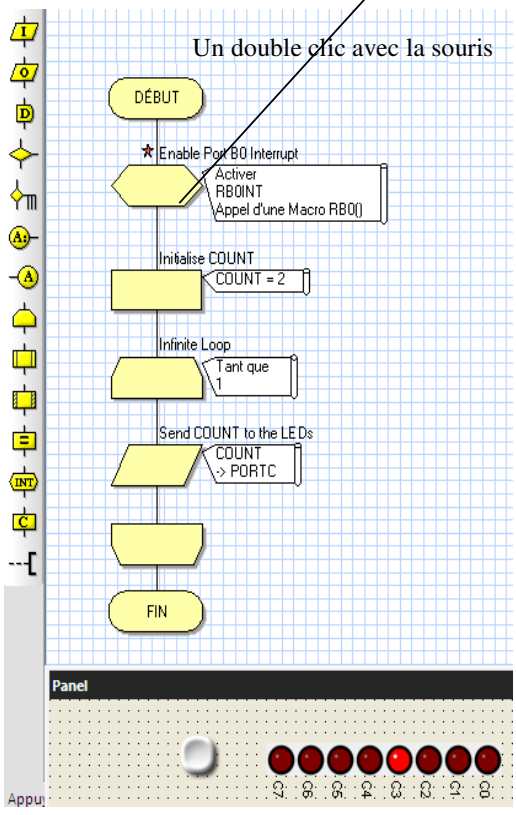


Figure 40: Configuration d'une interruption




```
 5  void FCM_Main()
 6  {
 7
 8      //Enable Port B0 Interrupt
 9      //Interruption: Activer RBOINT
10      cr_bit(option_reg,INTEDG);
11      st_bit(intcon,GIE);
12      st_bit(intcon, INTE);
13
14      //Initialise COUNT
15      //Calcul:
16      //   COUNT = 2
17      FCV_COUNT = 2;
18
19      //Infinite Loop
20      //Boucle: Tant que 1
21      while (1)
22      {
23
24          //Send COUNT to the LEDs
25          //Sortie: COUNT -> PORTC
26          trisc = 0x00;
27          portc = (FCV_COUNT);
28
29
30      }
31
32  }
```

Figure 41: Programme et organigramme principaux

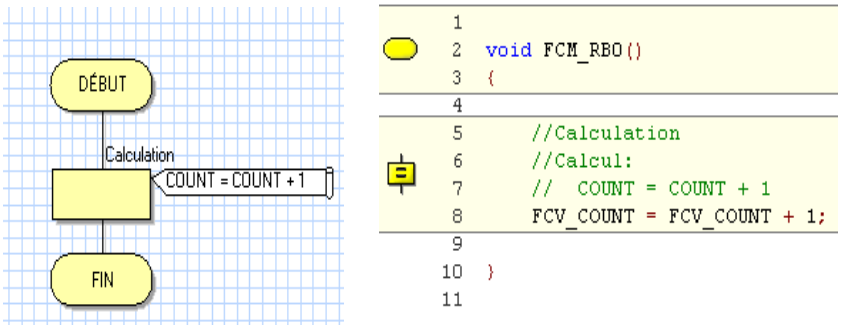


Figure 42: Sous programme et organigramme d'interruption

Exemple N°8 : Conversion analogique/numérique.

Le but de cet exemple est de faire une conversion analogique/numérique sur le port A (la broche RA0 du PIC16f877A) (Figure 44). Les LED rouges sont connectées aux ports B et C afin de vérifier la valeur numérique de la conversion codée sur 10 bits. Le programme correspondant obtenu après une opération de compilation avec le compilateur « MikroC Pro for PIC » de Mikroelektronika est le suivant :

```
unsigned int adc;
void main()
{ CMCON = 0x07; //désactiver le comparateur
 ADCON0 = 0xC1; // configuration du register ADCON0 (voir datasheet PIC16f87x, page 127)
 ADCON1 = 0x8E; //  configuration du register ADCON1 (voir datasheet PIC16f87x, page 128)
 TRISA = 0x01; // Broche  A0 en entrée
 TRISC = 0x00; // PORTC en sortie
 TRISB = 0x00;    // PORTB en sortie
 while(1)
 { adc = ADC_Read(0); // conversion A/N sur 10 bits – broche A0

 PORTB = adc;      // affectation des 8 bits du poids faible au PORTB
 PORTC = adc >> 8; // affectation des deux bits les plus significatives aux broches RC0, RC1
 delay_ms(200);//pause de 200 ms

 }
}
```

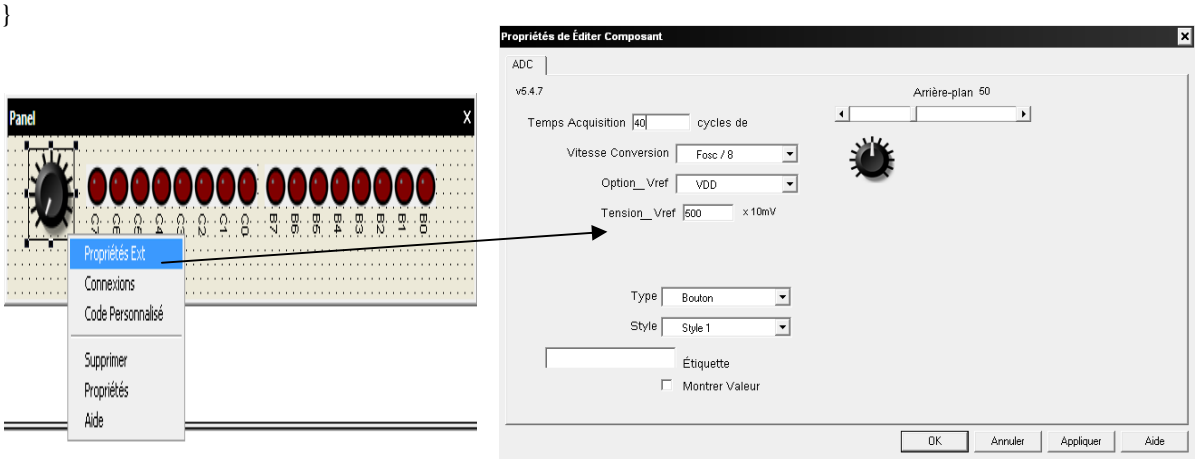


Figure 43: Configuration d'une entrée A/N

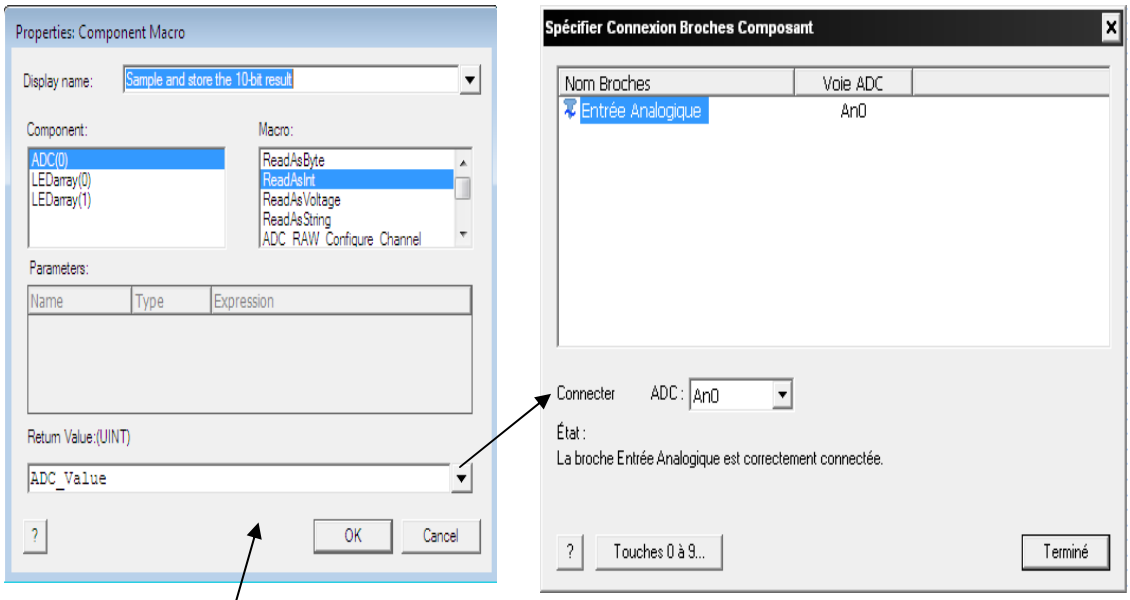


Figure 44: Choix de l'entrée A/N et du format de conversion

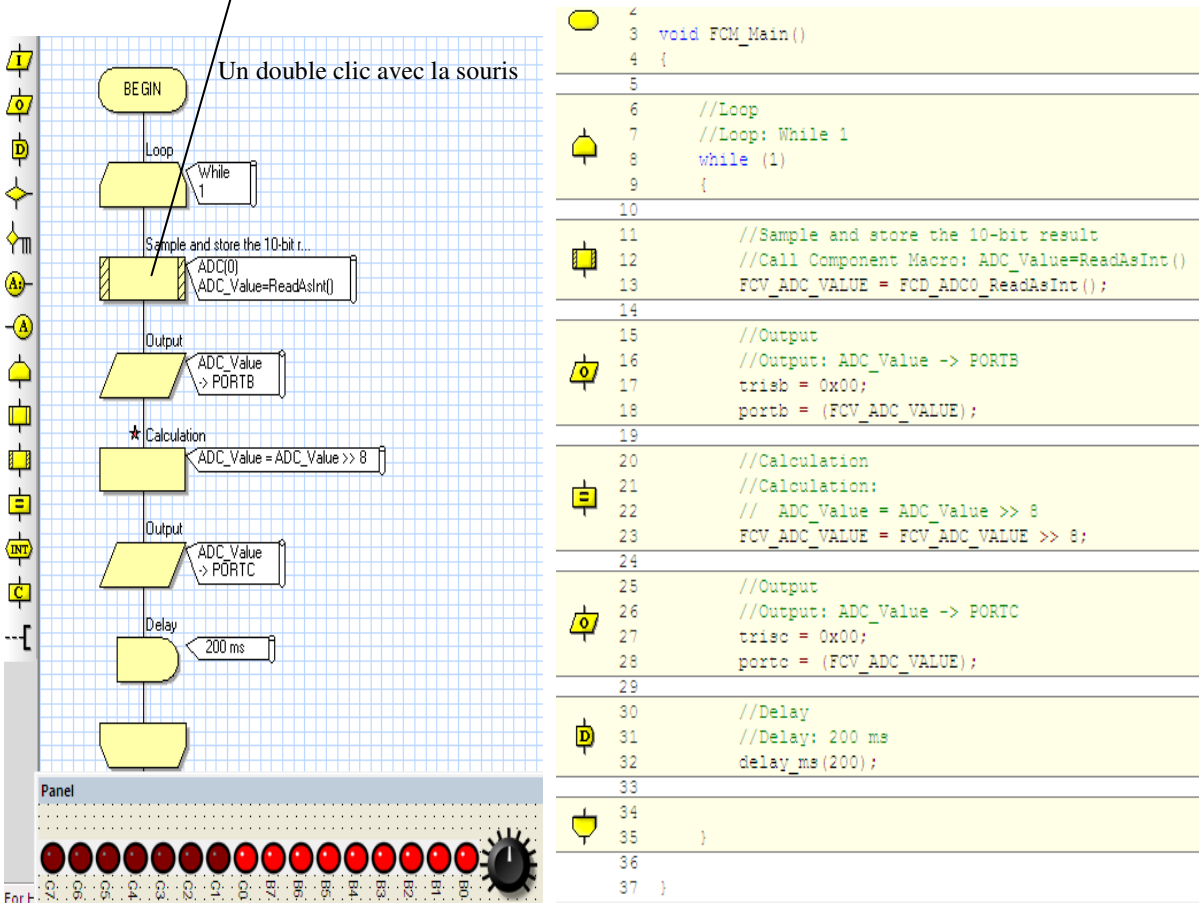


Figure 45: Programme de conversion A/N et d'affichage sur des LED

Exemple N°9 : Sortie Modulation de Largeur d'Impulsion (Pulse Width Modulation).
Le but de cet exemple est de générer un signal dont la largeur d'impulsion dépend de la valeur lue sur la broche AN0/RA0 du convertisseur analogique/numérique du PIC16f877A de 8MhZ. En cliquant sur l'icône verte représentant la PWM, alors la fenêtre (Figure 46) s'active permettant de configurer les paramètres du signal généré.

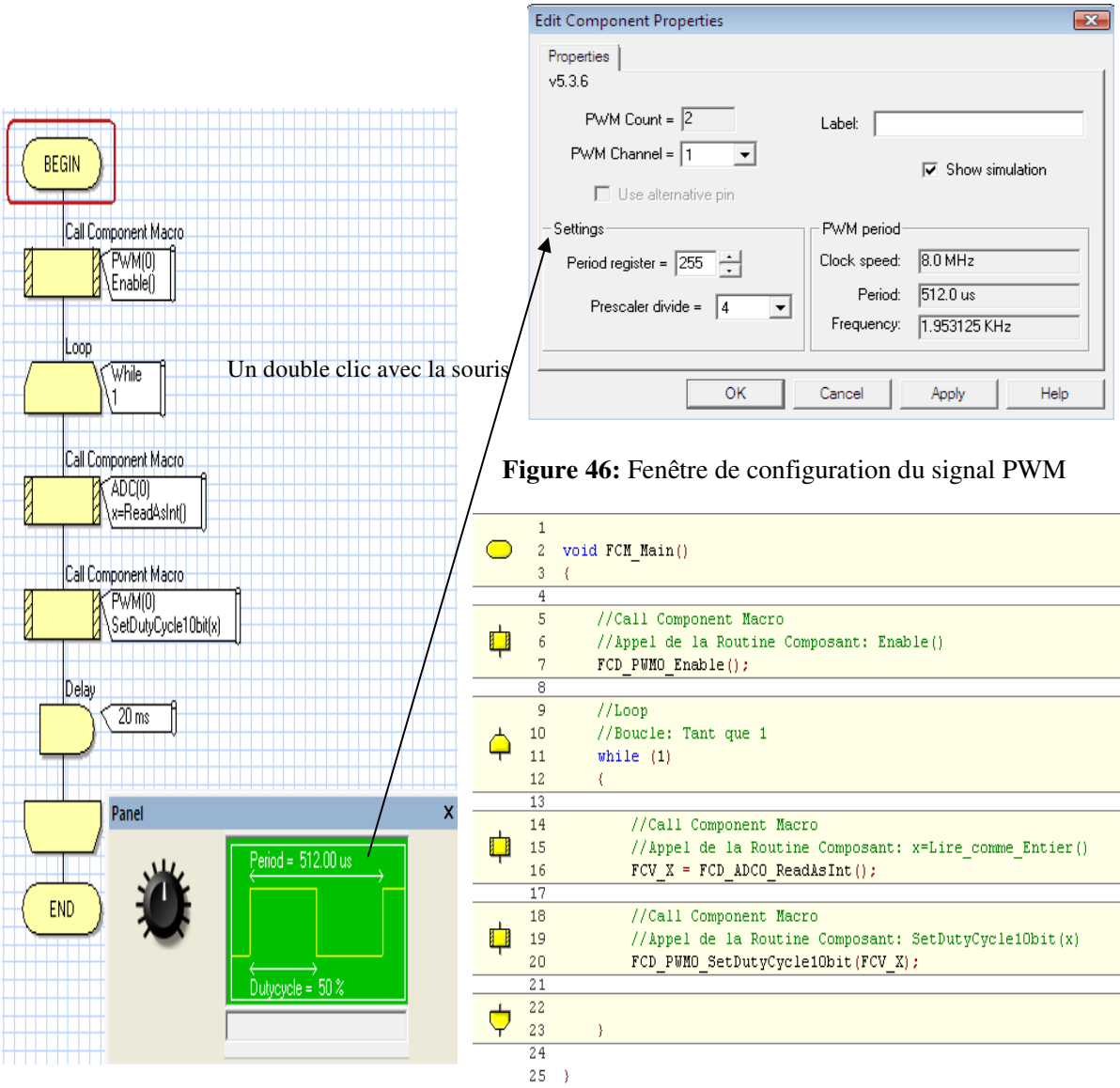


Figure 46: Fenêtre de configuration du signal PWM

```
1
2  void FCM_Main()
3  {
4
5      //Call Component Macro
6      //Appel de la Routine Composant: Enable()
7      FCD_PWM0_Enable();
8
9      //Loop
10     //Boucle: Tant que 1
11     while (1)
12     {
13
14         //Call Component Macro
15         //Appel de la Routine Composant: x=Lire_comme_Entier()
16         FCV_X = FCD_ADC0_ReadAsInt();
17
18         //Call Component Macro
19         //Appel de la Routine Composant: SetDutyCycle10bit(x)
20         FCD_PWM0_SetDutyCycle10bit(FCV_X);
21
22
23     }
24
25 }
```

Figure 47: Programme et organigramme de génération d'un signal PWM contrôlé par le convertisseur A/N

Exemple N°10: Compteur avec un afficheur à sept segments.
Le but de cet exemple est de créer un compteur à base de PIC16f877A de 8MhZ et d'un afficheur à sept segments. Une variable de type Byte est utilisé (count) afin de réaliser un compteur de 0 jusqu'à 9. Pour configurer l'afficheur, il suffit de faire un clic avec le bouton droit de la souris sur le composant (Figures 49 et 50).

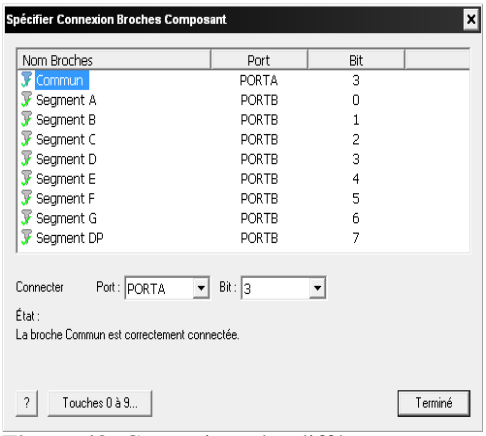
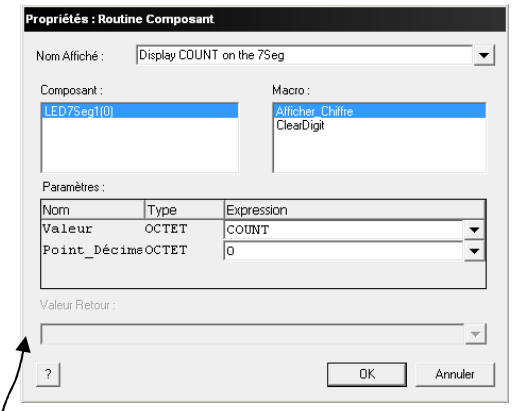


Figure 48: Connexions des différents segments au port B



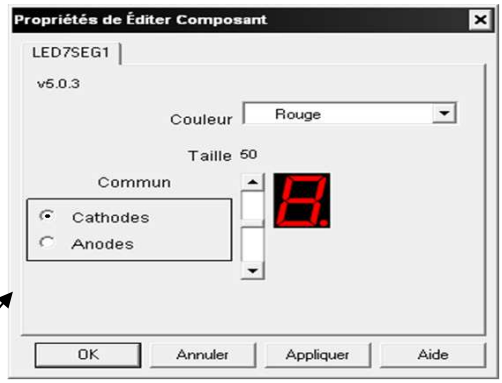


Figure 49: Propriétés de l'afficheur à sept segments

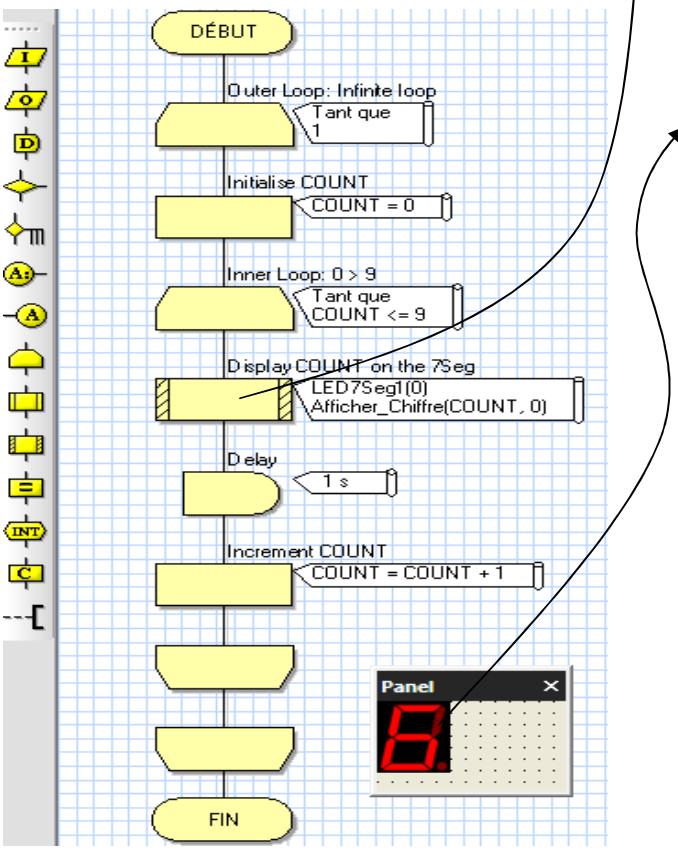


Figure 50: Organigramme du compteur à base d'un afficheur à sept segments

Exemple N°11: Appel d'une macro composant (Clavier).
Cet exemple donne un aperçu sur l'utilisation d'une macro composant de type clavier 4x4 (Figure 51). Ce dernier est branché au port D du PIC16f877A de 8MhZ (Figure 52). Le mot binaire correspondant à la touche appuyée est transmit directement au port C dont les broches RC0 jusqu'à RC3 sont connectées a des LED.

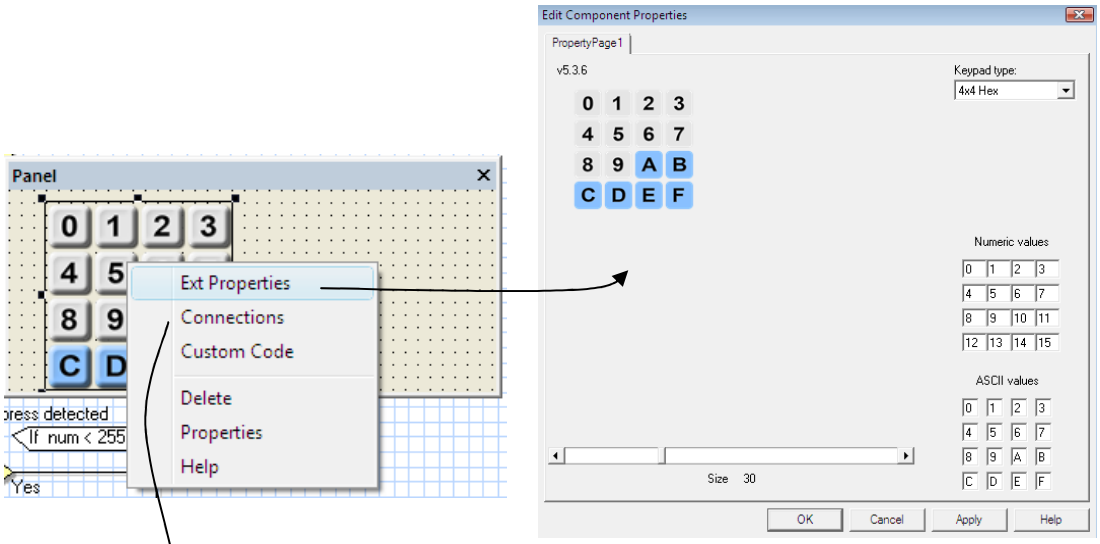


Figure 51: Configuration du clavier utilisé

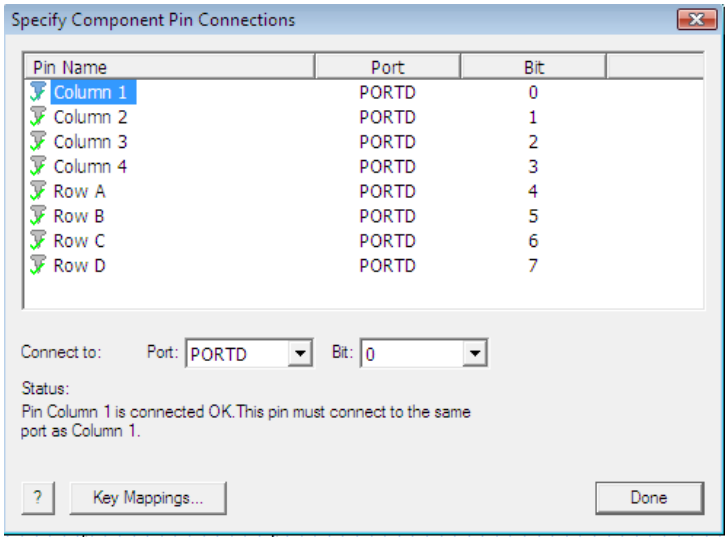


Figure 52: Configuration de la connexion du clavier au PIC16f877A

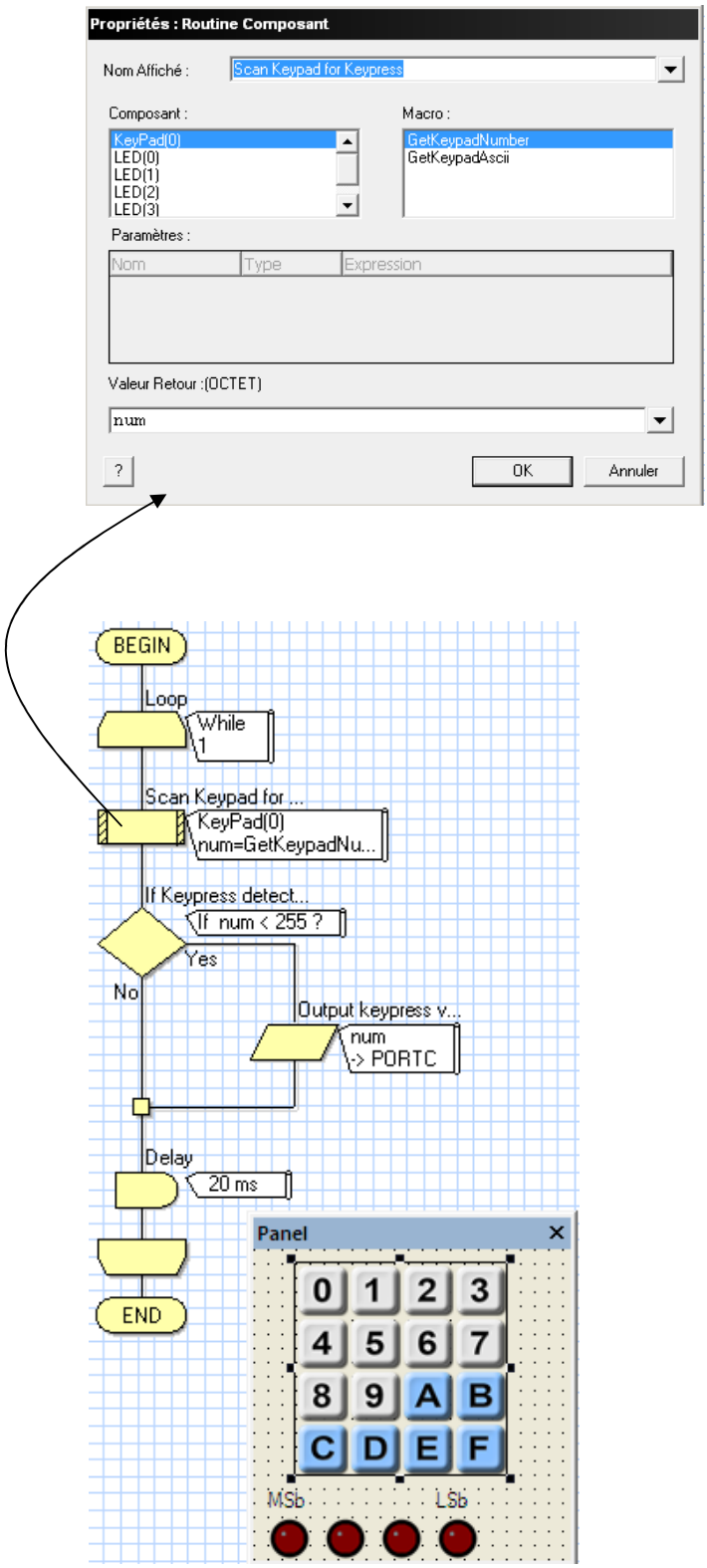


Figure 53: Organigramme de gestion d'un clavier 4x4 avec un affichage sur des LED

Exemple N°12: Appel d'une macro afficheur LCD (16x2)

Le but de cet exemple est de faire appel d'une macro composant de type LCD 16x2. Ce dernier est connectées port B du PIC16f877A de 8MhZ et fonctionne en mode 4bits. Flowcode permet de générer un fichier C après une compilation et en même temps il génère un fichier HEX qui peut être utilisé dans la simulation avec Proteus Isis ou directement chargé dans un PIC à travers un programmateur avec son logiciel (Figure 57).

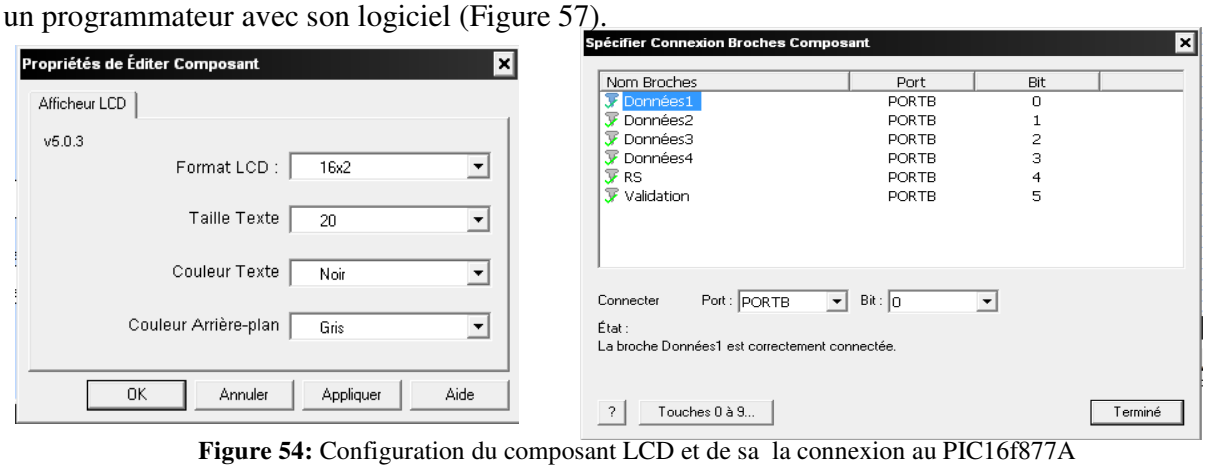


Figure 54: Configuration du composant LCD et de sa la connexion au PIC16f877A

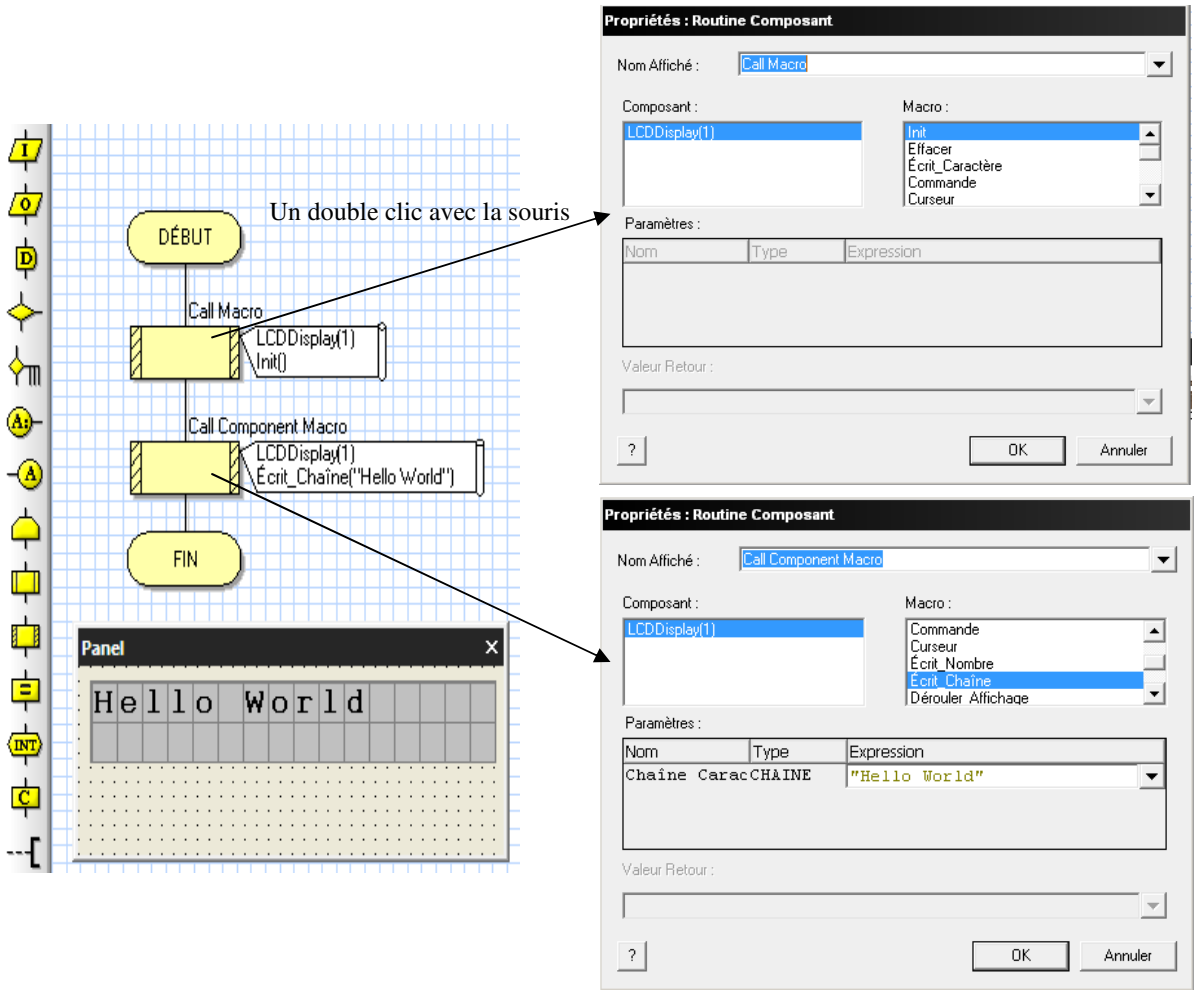


Figure 55: Organigramme d'affichage d'une chaine de caractères à base d'un LCD 16x2

```
1  //Using the LCD display
2
3  void FCM_Main()
4  {
5
6      //Call Macro
7      //Appel de la Routine Composant: Init()
8      FCD_LCDDisplay1_Start();
9
10     //Call Component Macro
11     //Appel de la Routine Composant: Écrit_Chaîne("Hello World")
12     FCD_LCDDisplay1_PrintString("Hello World", 11);
13
14 }
15
```

Figure 56: Programme C généré par Flowcode permettant de gérer l'affichage sur LCD 16x2

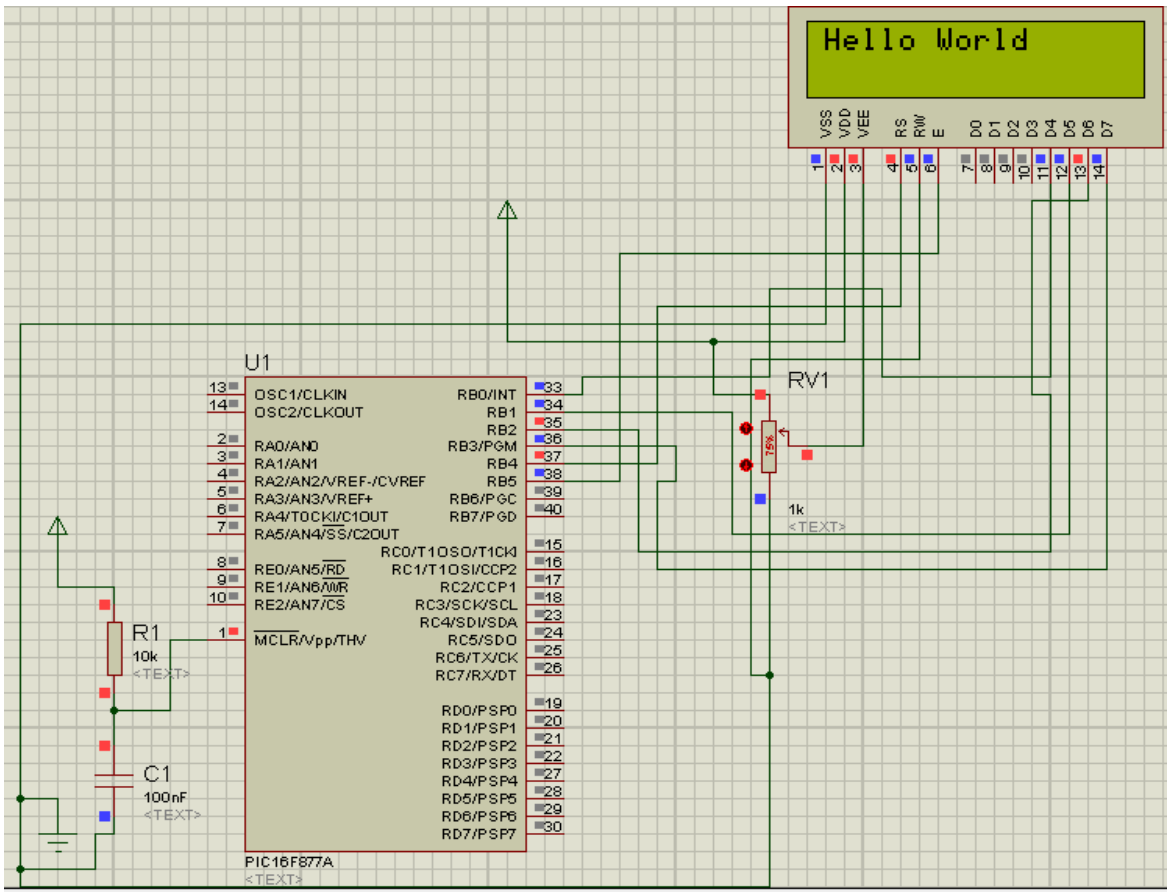


Figure 57: Simulation sur Proteus Isis V7.9 en exploitant le fichier HEX généré par Flowcode

14- Programmateur de PIC microcontrôleurs

La figure 58 représente le schéma du programmateur de PIC de type série 'JDM 2 PIC programmer'. Cette carte se connecte au port série d'un PC à travers un connecteur DB9 femelle et un ensemble de 05 files électriques. Le logiciel de programmation associé à cette carte est le WINPIC (Figure 59), nécessite l'installation de MPLAB pour les fichiers 'dev' contenant les définitions des différents microcontrôleurs. Donc, ce programmateur avec son logiciel ont été testés avec succès sur les PIC suivants : 16F84A, 16F877A, 18F458, EEPROM série 24LC16, et même les dsPIC : 30F2010, 30F4011.

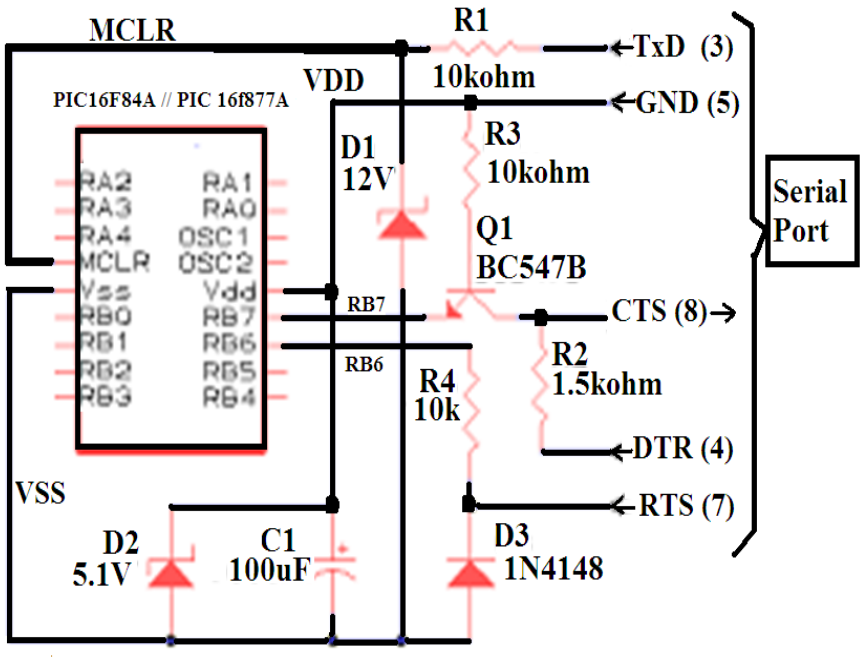


Figure 58: Schéma du programmateur de PIC "JDM (2)" connecté au port série d'un PC

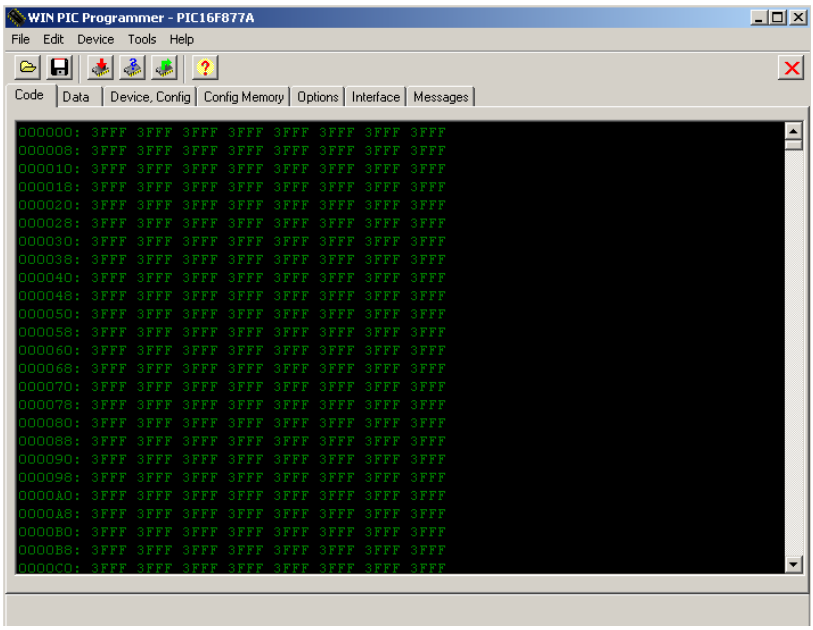


Figure 59: Logiciel de programmation de PIC "WINPIC Programmer"

Bibliographie

- Flowcode V5 Help file (EN). *https://www.matrixtsl.com.*

- FLOWCODE V4.2 Notice d'utilisation Préambule.
  https://bournigal.org/ressources/Notice_FLOWCODE_V4-2.pdf.

- FlowCode training videos
  https://www.youtube.com/playlist?list=PLQiU5-HoGR_Xwrv1WwbhG4au7RGeDs2ZN.

- Bert van Dam, "Microcontroller Systems Engineering", Publisher: Elektor Publishing, 2009.

- PIC16F877A - 8-bit PIC Microcontrollers
  http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf