FLOWCODE V4.3
Notice d'utilisation

Préambule :

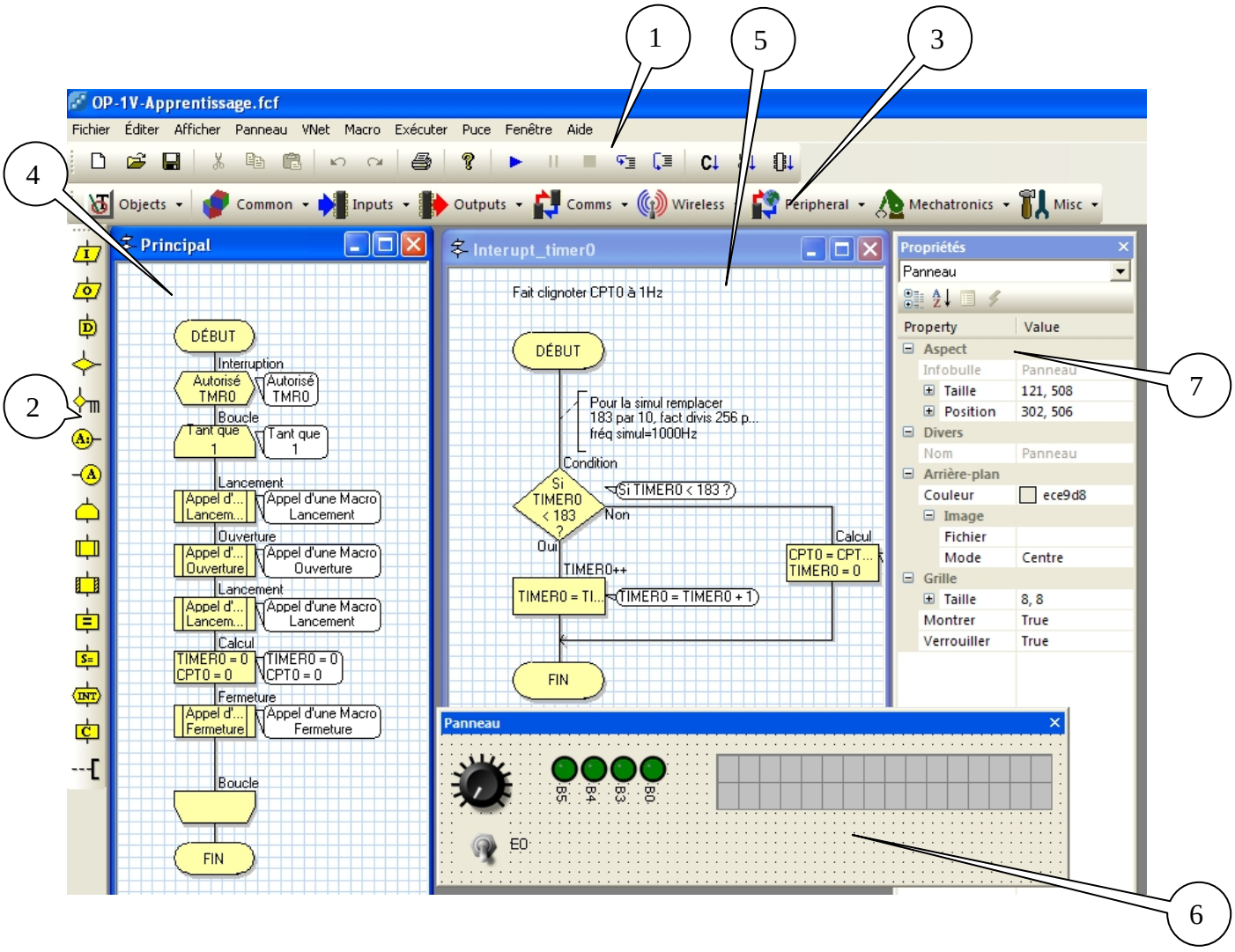
Flowcode est un logiciel de programmation graphique permettant, à partir de la saisie d'algorigrammes, de créer des programmes pour les microcontrôleurs de la famille des PICmicro® de Microchip.
Une fois l'algorigramme élaboré, Flowcode permet de simuler et visualiser le comportement du programme en découlant, avant de le traduire en langage C, de le compiler en hexadécimal et de le transférer dans le microcontrôleur cible.

Sommaire :

Présentation du logiciel

L'environnement Flowcode consiste en une aire de travail essentiellement graphique, dans laquelle s'affichent :

- trois barres d'outils : la barre d'outils de menus (1), les barres d'icônes de commandes (2) et des composants (3),
- l'algorigramme ou ordinogramme (4), qui se décompose en plusieurs fenêtres s'il comporte des sous-programmes (5),
- des fenêtres spécifiques pour montrer les composants attachés (regroupés dans un panneau de simulation) (6) et leurs propriétés (7), l'état du microcontrôleur, l'état des variables en mode simulation…



Nota :

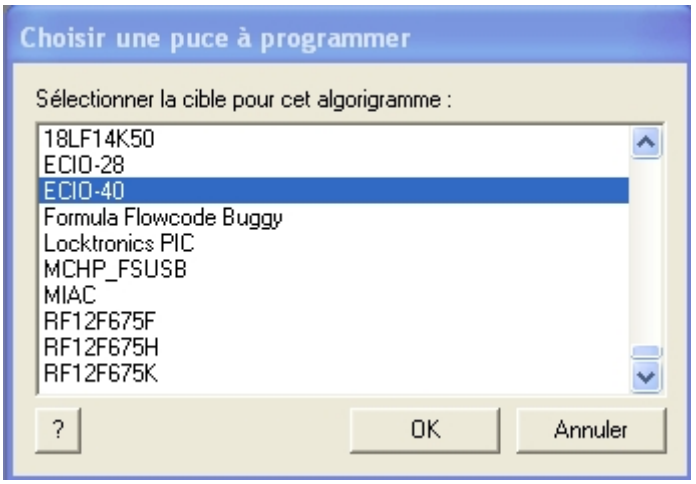
- Les symboles d'algorigrammes utilisés par Flowcode sont conformes à la norme internationale ISO 5807 (*), très proche de la norme française NF Z 61-100. La différence principale se situe au niveau des itérations (boucles répétitives) pour lesquelles la norme ISO utilise un symbole spécifique alors que la norme NF utilise le symbole de décision (alternative).
- Flowcode effectue une 1ère compilation de l'algorigramme en langage C, puis en assembleur, puis en langage machine. Ce passage par le langage C tranparaît dans la mise en œuvre de certaines fonctionnalités du logiciel.

ISO : international organization for standardization

Création d'un programme

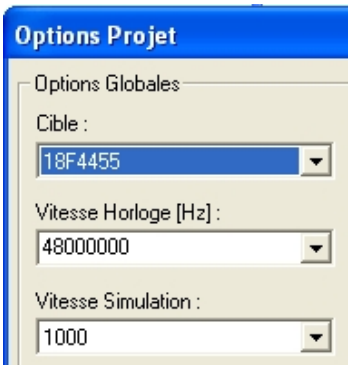
Pour créer un programme avec Flowcode il suffit de réaliser les étapes suivantes :

1. Lancer la création d'un nouvel algorigramme en spécifiant le microcontrôleur cible.



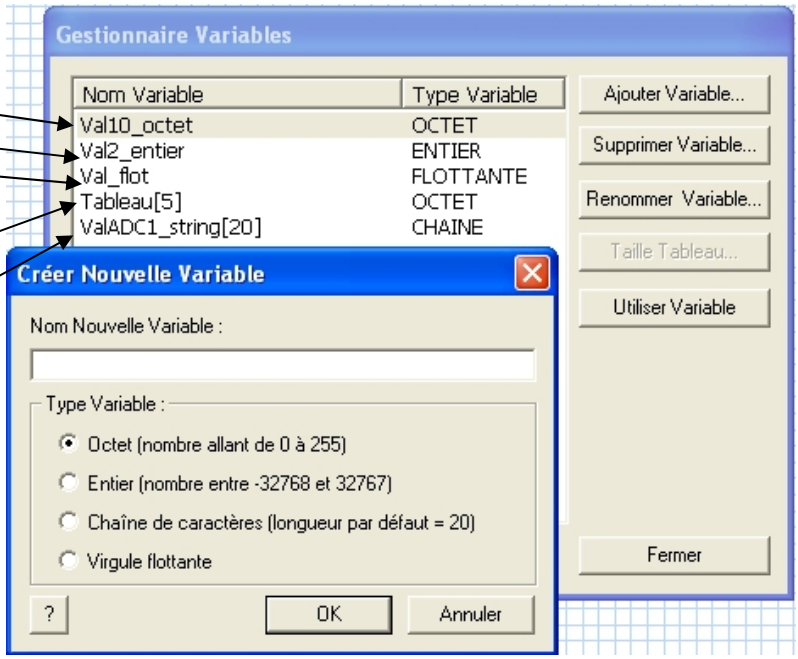
2. Préciser la vitesse d'horloge (fréquence) du microcontrôleur dans la boite de dialogue "Options Projet" du menu "Edition" afin que les réglages de temporisations ou d'interruption par timer soient précis. Spécifier également la vitesse de simulation (1000 préférentiellement).

   Note : Si le circuit cible est un ECIO la fréquence d'horloge est verrouillée sur 48MHz.



3. Établir la liste des variables qui seront utilisées dans l'algorigramme : menu Edition Variables…

   Flowcode V4 permet 5 types de variables :
   - Octet (nombre non signé à 8 bits)
   - Entier (nombre signé à 16 bits)
   - À virgule flottante (nombre signé à 32 bits soit 3.4 x 10^-38 à 3.4 x 10^+38)
   - Tableau d'octets ou d'entiers à une dimension (nombre de cellules entre crochets)
   - Chaîne de caractères (nombre de caractères entre crochets, 20 par défaut)



4. Sélectionner et faire glisser les icônes de la barre d'outils "Commandes" sur la fenêtre de saisie pour réaliser l'algorigramme : la 1ère fenêtre s'appelle "principal" (main), chaque macro (sous-programme) comporte une fenêtre de saisie.
   Note : pour optimiser l'espace, il est préférable de rendre flottante la fenêtre du panneau de simulation "Panel" (clic droit sur Panel, sélection de "Floating") puis d'ajuster sa taille.

5. Ajouter les périphériques externes nécessaires sur le panneau de simulation "Panel" en cliquant sur les boutons correspondants dans la barre d'outils "Composants", éditer leurs propriétés, spécifier leurs connexions au microcontrôleur et appeler/paramétrer les routines correspondant aux périphériques utilisés.

6. Faire exécuter la Simulation pour s'assurer que l'application se comporte conformément au cahier des charges.

7. Transférer l'application dans le microcontrôleur cible en compilant le programme en C, puis en l'assemblant et finalement en produisant et en transférant le code objet (Flowcode assure automatiquement ces opérations).

La barre d'ICÔNES DE COMMANDE

Propriétés de l'icône Entrée
Propriétés de l'icône Sortie
Propriétés de l'icône Pause
Propriétés de l'icône Décision (alternative simple ou complète)
Propriétés de l'icône Multi-décision (alternative généralisée)
Propriétés de l'icône Point de jonction
Propriétés de l'icône Boucle (itération)
Propriétés de l'icône Macro (sous-programme)
Propriétés de l'icône Routine composant
Propriétés de l'icône Calcul
Propriétés de l'icône Manipulation de caractères
Propriétés de l'icône Interruption
Propriétés de l'icône Code C
Propriétés de l'icône Commentaire

Pour éditer les Propriétés d'une icône, effectuer un double-clic sur l'icône dans l'algorigramme ou un clic droit sur l'icône : dans ce cas un menu contextuel s'affiche, sélectionner l'option Propriétés présente dans ce menu.

Ajouter une ICÔNE

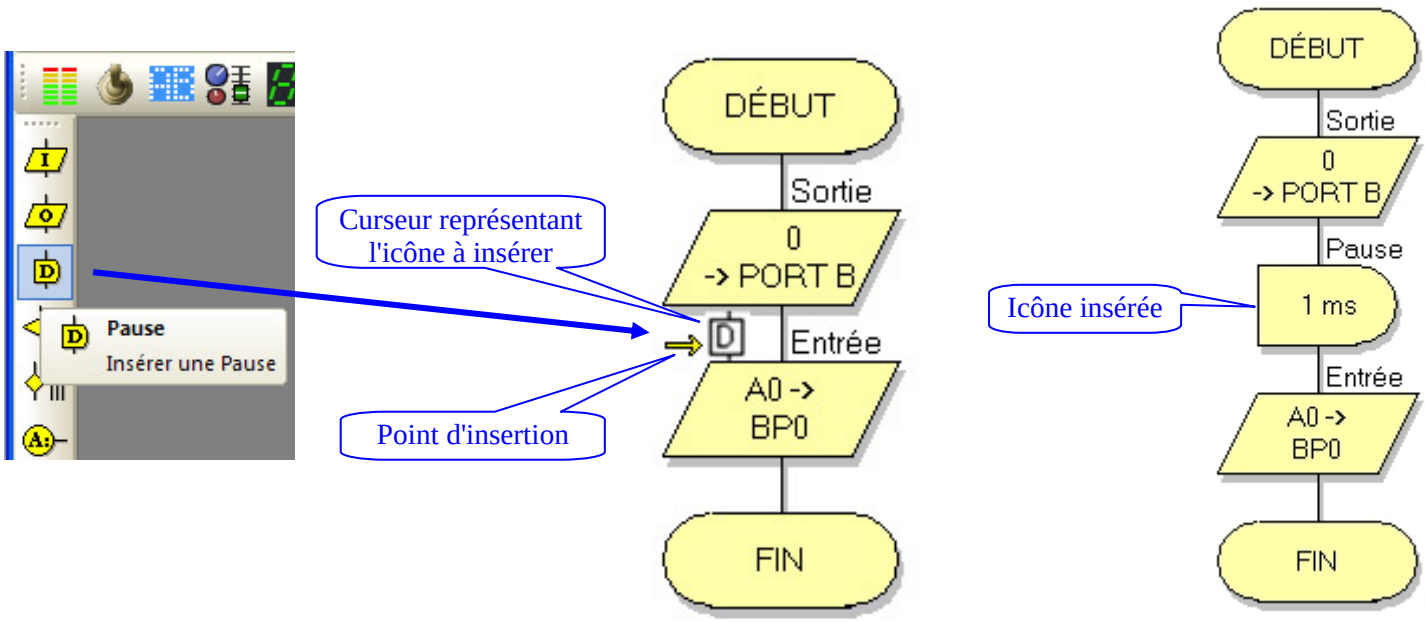
Pour ajouter une icône sur l'algorigramme :

- Clic gauche maintenu sur l'icône à insérer. Le curseur prend la forme d'une petite image de l'icône sélectionnée.
- Faire glisser l'icône dans la fenêtre active de l'algorigramme et relâcher la souris là où l'icône doit être insérée.

Nota :

- Quand vous déplacez la souris sur l'algorigramme, une petite flèche apparaît pour montrer où sera insérée l'icône quand le bouton de la souris est relâché. Ce point est identifié comme le point d'insertion.
- Dés que vous relâchez le bouton de la souris, l'icône s'inscrit dans l'algorigramme.
- Les fonctions classiques de Windows sont supportées : copier, couper, coller, déplacer…

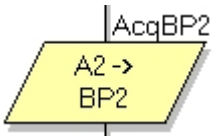
Exemple : insertion d'une temporisation (pause)



Pour éditer les Propriétés d'une icône, effectuer un double-clic sur l'icône dans l'algorigramme ou un clic droit sur l'icône : dans ce cas un menu contextuel s'affiche, sélectionner l'option Propriétés présente dans ce menu.

Propriétés de l'icône ENTRÉE

L'icône Entrée lit le port spécifié (ou certains bits seulement du port) et place le résultat dans la variable spécifiée.

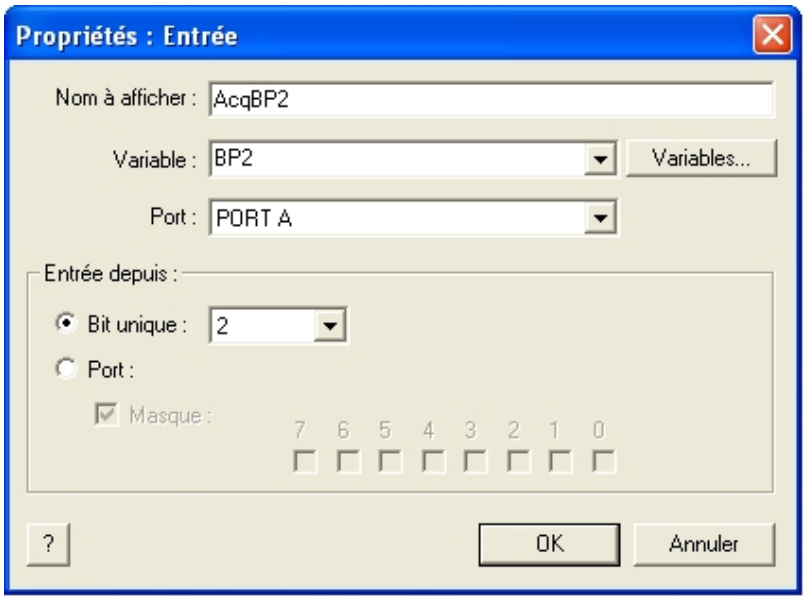


Nom à afficher

Le texte qui apparaîtra en haut et à droite de l'icône sur l'algorigramme.

Variable

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

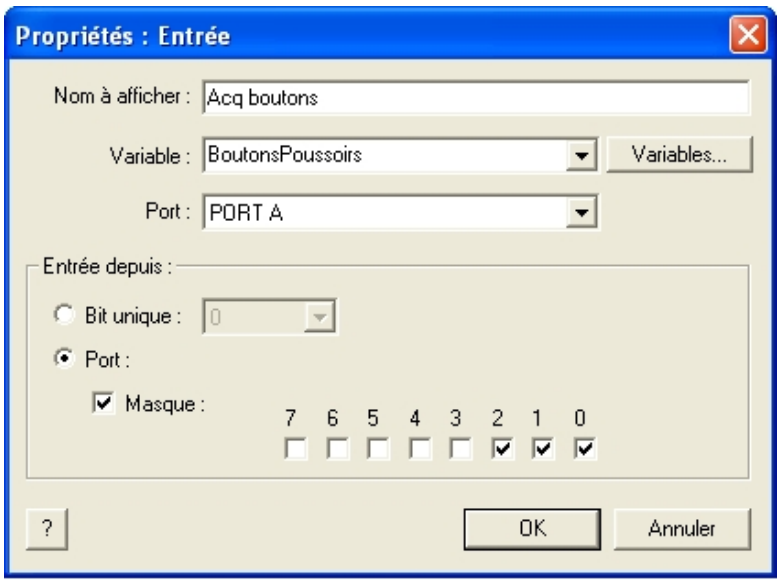
Utiliser cette option pour lire l'état d'un seul bit du port.
Dans l'exemple ci-contre l'état du bit 2 du PORT A (porta.2) est transféré dans la variable de type octet BP2. Cet octet n'a donc que 2 valeurs possibles 0 ou 1.

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.

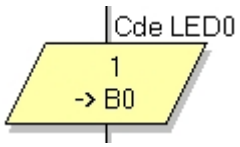
Masque

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.
Dans l'exemple ci-contre la variable de type octet BoutonsPoussoirs ne prend en compte que les bits 0, 1 et 2 du PORT A (les bits 3 à 7 de la variable prennent la valeur 0)



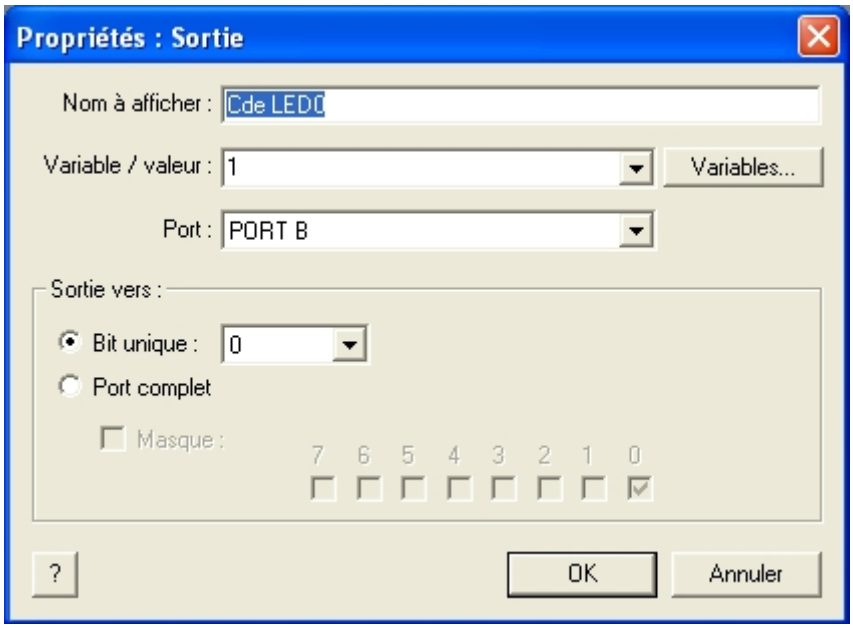
Propriétés de l'icône SORTIE

L'icône Sortie 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.



Nom à afficher
Le texte qui apparaîtra en haut et à droite de l'icône sur l'algorigramme.

Variable ou valeur
Sélectionner la variable ou la valeur numérique au format décimal (type par défaut), hexadécimal (précédée par 0x) ou binaire (précédé par 0b) que vous souhaitez écrire dans ce port.



Bouton Variables…
Ce bouton ouvre la boite de dialogue Variables permettant de sélectionner une variable existante ou d'en créer une nouvelle.

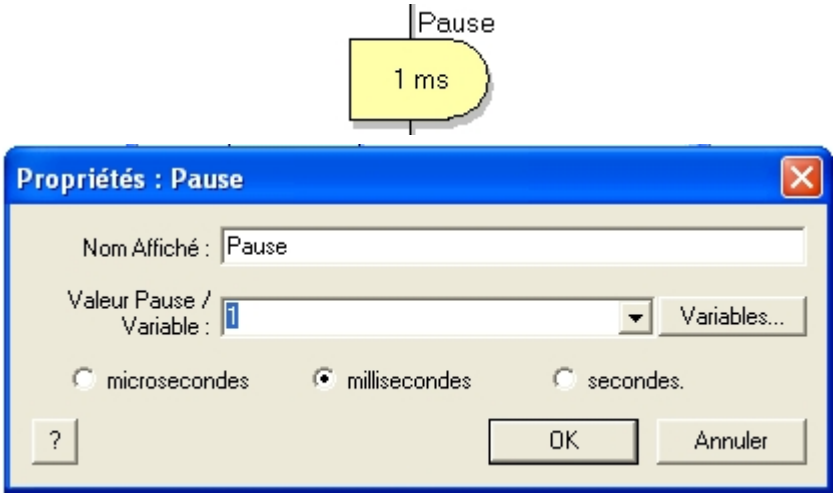
Port
Le sélectionner depuis la liste des ports disponibles sur le PICmicro à programmer

Sortie vers Bit unique
Utiliser cette option pour écrire dans un seul bit du port.
Si une valeur différente de zéro est écrite dans ce bit alors le bit est mis à 1, sinon le bit est mis à 0.

Sortie vers Port complet
Décocher masque. Utiliser cette option pour écrire la valeur ou la variable dans le port entier.

Masque
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.
Dans l'exemple ci-contre, seuls les bits 0 à 4 du PORT B sont mis à 1 lors de l'écriture de la valeur FF dans le port puisque les bits 5 à 7 sont masqués.

Propriétés de l'icône PAUSE



L'icône Pause permet d'insérer des temporisations dans votre programme et d'en ralentir l'exécution.
Attention : 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.

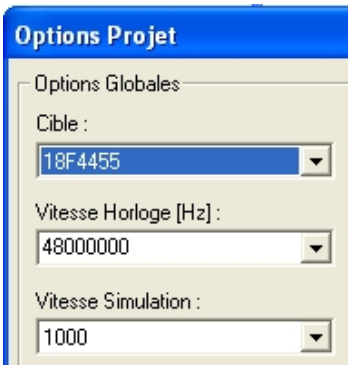
Nom à afficher
Le texte qui apparaîtra en haut et à droite de l'icône sur l'algorigramme.

Valeur ou variable Pause
Ceci correspond à la longueur de la pause que vous voulez créer sous en donnant directement la valeur soit en liant la valeur à une variable (→ temporisation paramétrique)

Bouton Variables
Ce bouton ouvre la boite de dialogue Variables permettant de sélectionner une variable existante ou d'en créer une nouvelle.

Options Microsecondes/Millisecondes/Secondes
Les pauses (ou temporisations) peuvent être exprimées en microsecondes, millisecondes ou secondes. Lorsque la simulation rencontre une pause exprimée en secondes, une boite de dialogue apparaît montrant le décompte du temps. Le bouton Annuler de cette fenêtre de progression permet de poursuivre l'exécution de l'algorigramme sans avoir à attendre que le temps soit complètement écoulé.

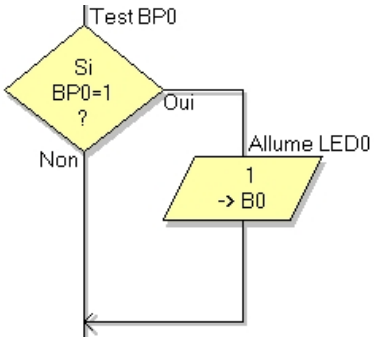
Pour que Flowcode puisse correctement programmer votre PICmicro avec des réglages de temporisations précis, vous devez préciser la vitesse d'horloge (fréquence) de votre PICmicro (boite de dialogue "Options Projet" du menu "Edition").



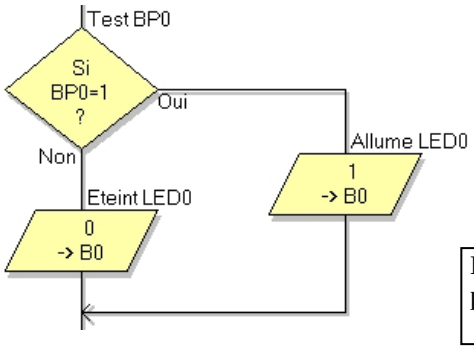
Propriétés de l'icône DÉCISION (alternative)

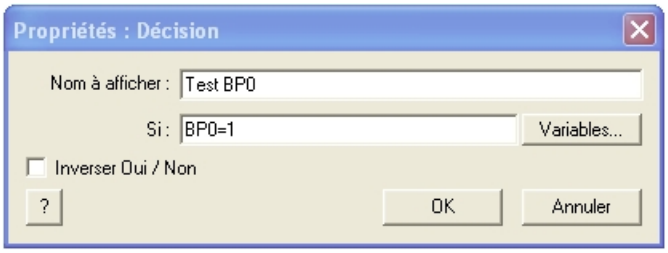
L'icône de Décision, appelée aussi Alternative, permet de tester une condition et d'effectuer certains traitements en fonction du résultat du test.

Alternative simple



Alternative complète (double)





Nota : si le test porte sur un bit, comme dans l'exemple, on peut écrire Si BP0 pour tester si BP0 actionné ou
Si !BP0 pour tester si BP0 relâché

Nom à afficher : Texte à afficher sur l'algorigramme 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.
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 comme :

(,)	- 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, ET, OU, OU Exclusif (opération bit à bit)
! && ||	- NON, ET, OU (opérations sur octet(s), le résultat vaut 0 ou 1)

Les valeurs numériques peuvent être écrites au format décimal (type par défaut), hexadécimal (précédée par 0x) ou binaire (précédé par 0b). Exemples : 255 ou 0xFF ou 0b01010101.

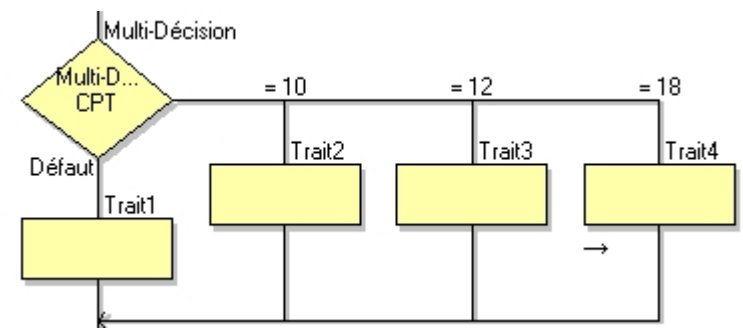
Bouton Variables : Ce bouton ouvre la boite de dialogue Variables permettant de sélectionner une variable existante ou d'en créer une nouvelle.

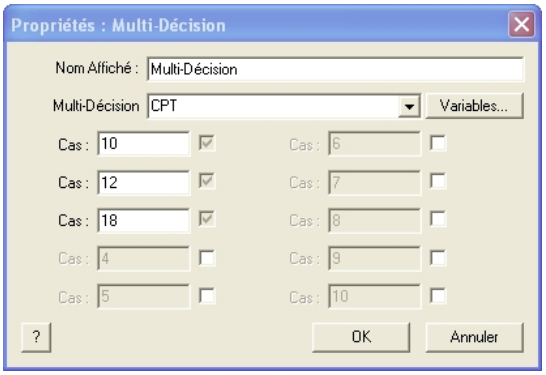
Inverser Oui et Non
Normalement la branche correspondant à "Oui" part sur la droite de l'icône de Décision et la branche correspondant au 'Non' continue tout droit dans l'algorigramme. Cocher cette option pour inverser les deux branches.

Valeurs logiques : Flowcode considère zéro comme FAUX et toute autre valeur différente de zéro comme VRAI.

Propriétés de l'icône MULTI-DÉCISION (alternative généralisée)

Dans l'exemple ci-dessous : si CPT=10 le traitement Trait2 s'effectue, si CPT=12 le traitement Trait3 s'effectue, si CPT=18 le traitement Trait4 s'effectue et dans tous les autres cas le traitement par défaut cad Trait1 s'effectue.
Nota : les valeurs peuvent écrites au format décimal, hexadécimal ou binaire.



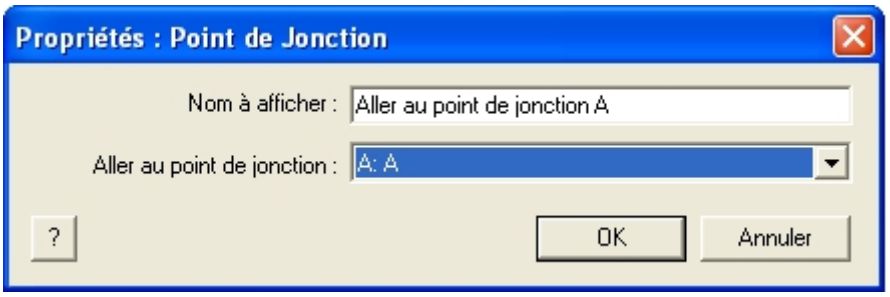


Propriétés de l'icône POINT DE JONCTION (saut inconditionnel)

Les icônes de jonction sont utilisées pour "sauter" d'un point de l'algorigramme à un autre.
Quand l'algorigramme 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 :

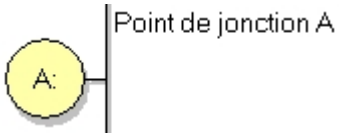
- le premier est le **point de saut** cad le point dans l'algorigramme à partir duquel il faut effectuer le saut.

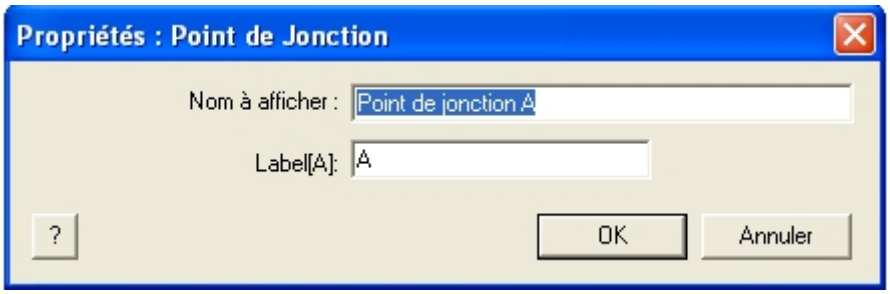


Nom à afficher
Texte à afficher sur l'algorigramme en haut et à droite de l'icône.

Aller au point de jonction
Sélectionner le point de jonction auquel vous voulez aller. Cette option n'est pas disponible si l'icône correspond à la définition d'un point de jonction plutôt qu'à un point de saut.

- le second est le **point de jonction** cad le point dans l'algorigramme indiquant à quel endroit se rendre. Le point de saut et le point de jonction partagent une lettre de jonction (label) – dans ce cas, la lettre 'A'.



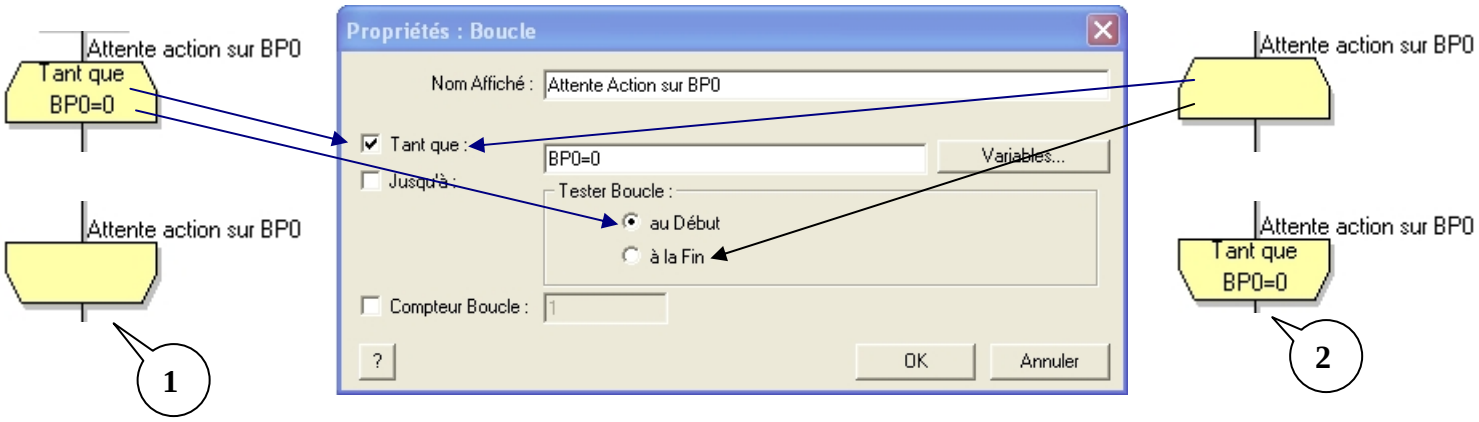


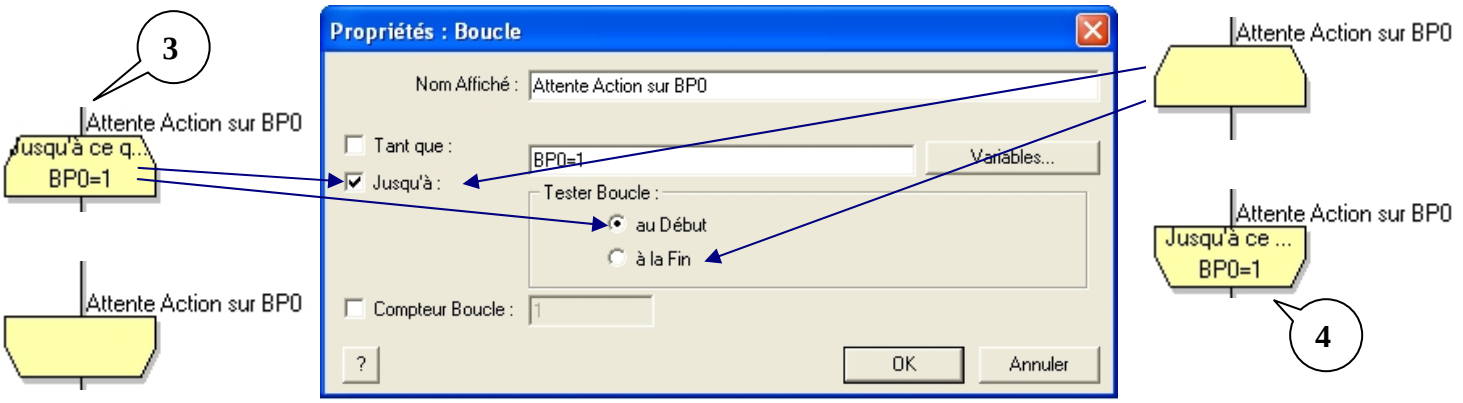
Nota :

Plusieurs points de saut peuvent faire référence à un même point de jonction.

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

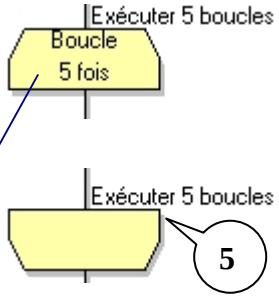
Propriétés de l'icône BOUCLE (itération)

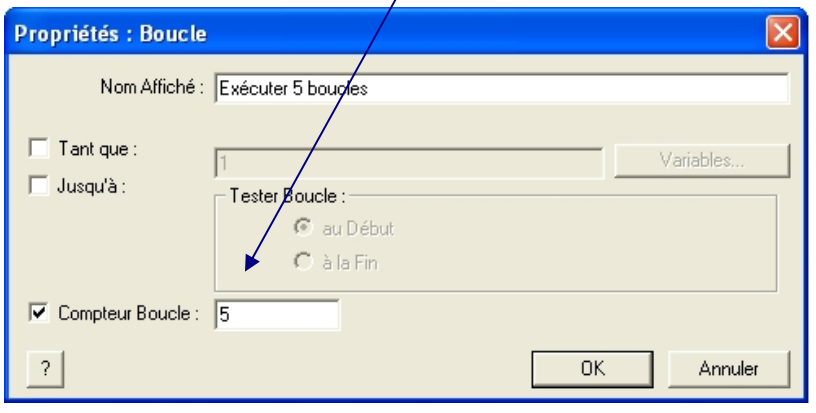




Les icônes Boucle sont utilisées pour mettre en œuvre des structures itératives (répétitives). Cinq types d'itérations sont réalisables :

- La boucle "Tant que … Faire…" (test de la boucle au début) **(1)**
- La boucle "Faire… Tant que …" (test de la boucle à la fin) **(2)**
- La boucle "Répéter… Jusqu'à …" (test de la boucle au début) **(3)**
- La boucle "Jusqu'à… Répéter …" (test de la boucle à la fin) **(4)**
- La répétition de boucle un nombre de fois spécifié (de 1 à 255) **(5)**





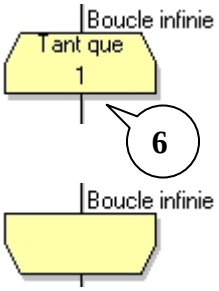
Nom à afficher
Texte à afficher sur l'algorigramme en haut et à droite de l'icône.

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.

Tester la boucle :
Cette option permet de spécifier si la boucle doit être testée au début ou à la fin de la boucle.



Cas particulier : Boucle infinie
Il arrive qu'une tâche soit répétée à l'infini (par ex scrutation). Une façon pratique d'obtenir ce fonctionnement est d'utiliser une boucle infinie. Tester une condition "Toujours Vrai" cad "1" fera que la boucle sera répétée indéfiniment **(6)**.

Les macros sont des portions de code **réutilisables** dans un projet.
Les macros permettent de diviser les tâches complexes en blocs de code élémentaires que l'on peut importer et exporter.

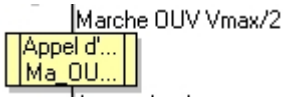
Les macros dans Flowcode sont scindées en deux catégories : les macros (de logiciel) et les routines composant (de matériel).

Propriétés de l'icône ROUTINE COMPOSANT

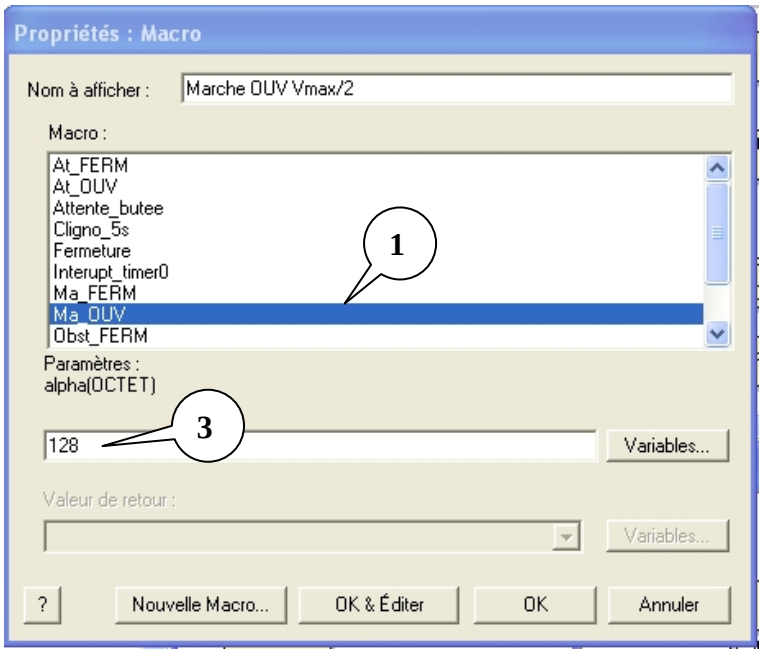
Les routines composant sont des macros prédéfinies qui accompagnent les composants fournis par Flowcode. Par exemple, les macros LCD permettent d'afficher des caractères alphanumériques sur l'écran LCD.
Une routine Composant fonctionne uniquement avec un composant déterminé.
L'icône de la routine Composant se reconnaît aux bandes hachurées sur le bord extérieur.
Certaines routines composant sont décrites plus loin.

Propriétés de l'icône MACRO

L'utilisateur peut concevoir et écrire ses propres macros de la façon décrite ici.
L'utilisateur peut exporter et importer les macros pour les regrouper en bibliothèques de tâches courantes.
La bordure extérieure de l'icône des macros logicielles est claire et sans hachures.



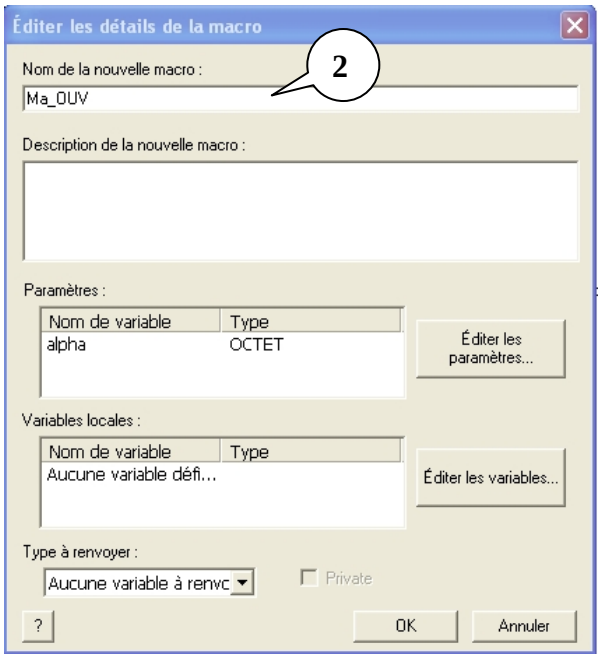
Un double clic sur l'icône ouvre le menu de Propriétés des macros **(1)** et permet à l'utilisateur de sélectionner ou d'ajouter des macros.



Pour réutiliser une macro existante, sélectionnez dans la liste la macro à utiliser.

Sinon 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 **(2)**.
Pour confirmer l'ajout et ouvrir la fenêtre d'édition de l'algorigramme de la nouvelle macro, cliquez sur OK & Éditer Macro.

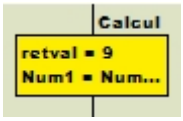
Paramètres **(3)**
Si la macro a besoin de paramètres, ils doivent être introduits dans ce champ. Il peut s'agir de valeurs numériques ou de variables existantes. Chaque variable ou valeur doit être séparée par une virgule dans la liste.
Le détail des paramètres affichera le type de chaque paramètre. Pour être acceptés, les paramètres doivent être du type déclaré.
Notez qu'un jeu complet de paramètres doit être fourni.

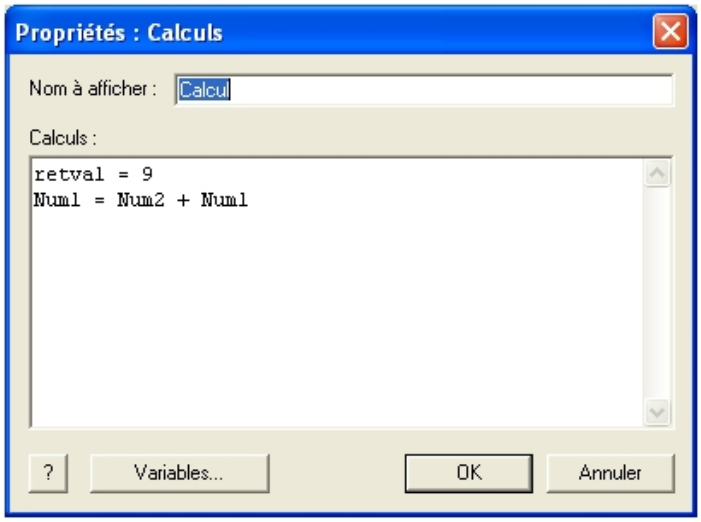


Le type de variable pour la valeur de retour sera affiché. En effet, il faut utiliser une variable du type adéquat pour accueillir la valeur de retour.

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.





Nom à afficher
Texte à afficher sur l'algorigramme 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 nombres, 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)

Les valeurs numériques peuvent être écrites au format décimal (type par défaut), hexadécimal (précédée par 0x) ou binaire (précédé par 0b). Exemples : 255 ou 0xFF ou 0b01010101.

A partir du moment où les variables ont été préalablement définies, toutes les lignes suivantes sont des lignes de calculs parfaitement valides :

```
TEMPO = TEMPO + 1
TEMPO = (MA_VARIABLE + 3) * 3
BITSUIVANT = DERNIERBIT >> 2 AND MASK
AETB = PORT_A AND PORT_B
```

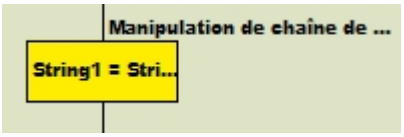
Bouton Variables
Ce bouton ouvre la boite de dialogue des variables afin de choisir une variable existante ou d'en créer une nouvelle.

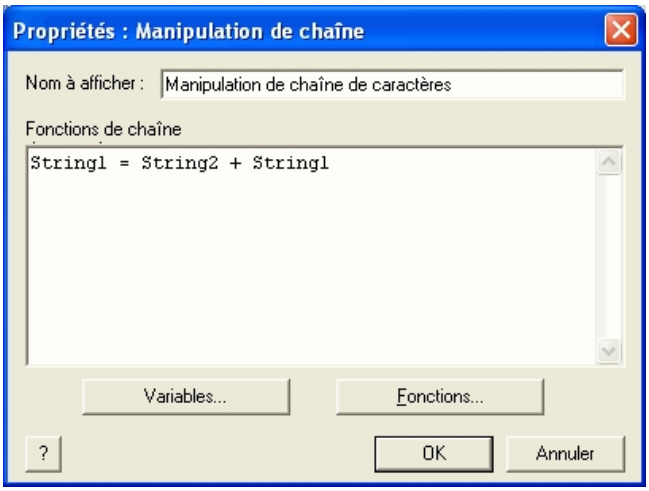
Fonctions spécifiques (Flowcode PIC seulement)
Flowcode inclut un certain nombre de fonctions mathématiques :

float = fadd(float, float)	- Additionne deux variables de type "virgule flottante" (format signé 32 bits)
float = fsub(float, float)	- Soustrait deux variables de type "virgule flottante"
float = fmul(float, float)	- Multiplie deux variables de type "virgule flottante"
float = fdiv(float, float)	- Divise deux variables de type "virgule flottante"
float = fmod(float, float)	- Donne le modulo de deux variables de type "virgule flottante"
byte = isinf(float)	- Teste si la variables de type "virgule flottante" est infinie
byte = isnan(float)	- Teste si la variables de type "virgule flottante" n'est pas un nombre
byte = float_eq(float, float)	- Teste l'égalité de deux variables de type "virgule flottante"
byte = float_ge(float, float)	- Teste si les deux variables de type "virgule flottante" sont supérieures ou égales
byte = float_gt(float, float)	- Teste si les deux variables de type "virgule flottante" sont strictement supérieures
byte = float_le(float, float)	- Teste si les deux variables de type "virgule flottante" sont inférieures ou égales
byte = float_lt(float, float)	- Teste si les deux variables de type "virgule flottante" sont strictement inférieures
int = random()	- Génère un nombre aléatoire compris entre -32768 et 32767

Propriétés de l'icône MANIPULATION DE CARACTÈRES

La fonction de manipulation de chaîne de caractères permet à l'utilisateur d'agir sur des chaînes de la même manière que la fonction de calcul permet d'agir sur des variables numériques.





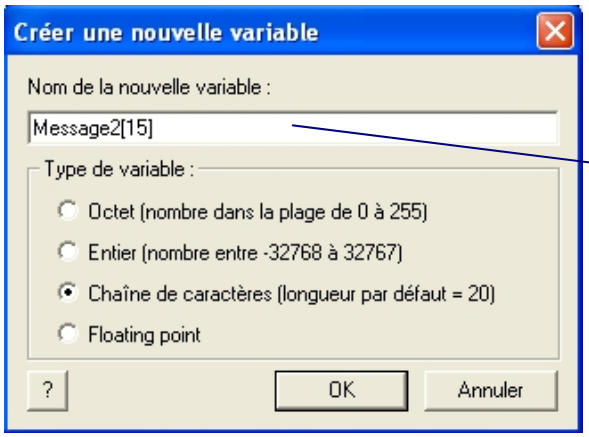
On introduit les opérations à effectuer sur les chaînes de caractères dans la zone de saisie "Fonctions de chaîne". Les boutons "Variables…" et "Fonctions…" permettent à l'utilisateur d'ajouter des éléments dans la zone de saisie.

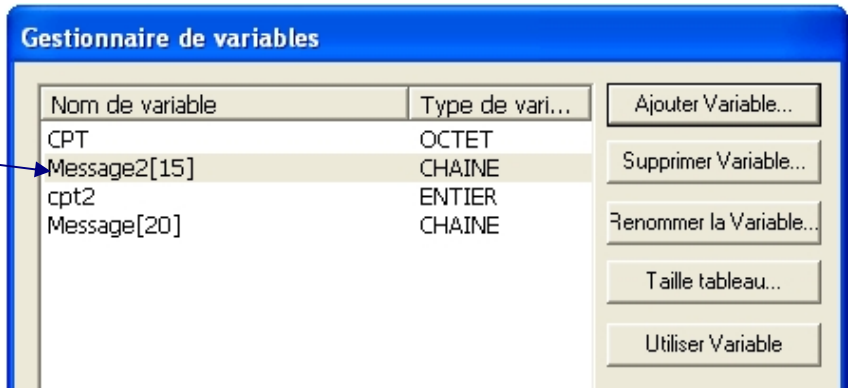
Chaînes

Les chaînes sont des rangées d'octets qui représentent des caractères ASCII.
Une chaîne est définie par son nom de chaîne accolé à son nombre de caractères entre crochets.
MyString[24] par exemple est une chaîne appelée MyString et dont la longueur est de 24 caractères.

Créer une chaîne

On crée les chaînes dans l'écran de Variables au milieu des autres variables.
Pour créer une chaîne, on saisit le nom de la chaîne et l'on pointe l'option retenue.
Pour fixer la longueur de la rangée, on l'introduit entre crochets.
À défaut de spécification, la longueur de la chaîne sera de 20 caractères.





Changer la longueur de la chaîne demande d'ouvrir la fenêtre des variables, de sélectionner la chaîne à mettre à dimension et de sélectionner Renommer la variable. Il n'y a plus qu'à introduire la nouvelle longueur.
Remarquez qu'il faut modifier le code de votre programme pour qu'il prenne en compte la nouvelle longueur.

Fonctions de manipulation de chaîne de caractères

Les fonctions de manipulation de chaîne permettent d'éditer, modifier et examiner les chaînes.
Cliquer sur une fonction ajoute le code de base dans la fenêtre de la boîte d'édition où l'on pourra l'éditer avec les variables nécessaires.

Exemples de chaînes pour illustrer les fonctions :
Str1[20] = "Hello "
Str2[10] = "World"
TestStr[20]

= *Remplace le contenu de la chaîne. Si la nouvelle chaîne est plus longue, les caractères en trop seront perdus.*

TestStr = Str2 *TestStr est devenu* « World ».

+ *Opère la concaténation (association) de deux chaînes dans l'ordre indiqué. Si la chaîne résultante est plus longue que celle qui reçoit la concaténation, les caractères en trop seront perdus.*

TestStr = Str1 + Str2 *TestStr est devenu* « Hello World ».

ToString$(valeur) *Change la valeur numérique d'une chaîne.*

TestStr = ToString$(1234) *TestStr est devenu* « 1234 ».

ToUpper$(chaîne) *Transforme toutes les lettres en capitales.*

TestStr = ToUpper$(Str1) *TestStr est devenu* « HELLO ».

ToLower$(chaîne) *Transforme toutes les lettres en minuscules.*

TestStr = ToLower$(Str1) *TestStr est devenu* « hello ».

Length$(chaîne) *Retrouve la longueur de la chaîne. Ce n'est pas la dimension de la rangée, mais le nombre de caractères avant de rencontrer un blanc.*

RetVal = Length$(Str1) *RetVal vaut* 6.

Note : la dimension de Str1 est de 20, mais la chaîne ne compte actuellement que 6 caractères, donc la réponse est 6.

Left$(chaîne, longueur) *Constitue, à partir de la chaîne désignée, un morceau de chaîne de la longueur spécifiée en commençant par la gauche. Si la longueur du morceau de chaîne est plus grande que celle de la variable de destination, les caractères en trop seront perdus.*

TestStr = Left$(Str1, 3) *TestStr devient ainsi* « Hel ».

Right$(chaîne, longueur) *Constitue, à partir de la chaîne désignée, un morceau de chaîne de la longueur spécifiée en commençant par la droite. Si la longueur du morceau de chaîne est plus grande que celle de la variable de destination, les caractères en trop seront perdus.*

TestStr = Right$(Str1, 3) *TestStr est à présent* « lo ».

Mid$(chaîne, début, longueur) *Constitue, à partir de la chaîne désignée et en commençant à « début », un morceau de chaîne de la longueur spécifiée. Si la longueur du morceau de chaîne est plus grande que celle de la variable de destination, les caractères en trop seront perdus.*

TestStr = Mid$(Str1, 2, 3) *TestStr contient alors* « llo ».

Compare$(chaîne1, chaîne2, compare_type) *Compare la chaîne1 à la chaîne2 et renvoie un octet dont la valeur est déterminée par les règles suivantes :*

0 si les chaînes sont identiques
1 si chaîne1 > chaîne2
255 si chaîne2 > chaîne1

Le troisième paramètre, compare_type, détermine si la comparaison *doit tenir compte de la casse ou non. Ce paramètre peut prendre deux valeurs :*

0 = sensible à la casse
1 = indifférent à la casse.

Exemples
Str1 = "ABC"
Str2 = "abc"
RetVal = Compare$(Str1, Str2, 0) *RetVal vaut 255 du fait que Str2 apparaît plus loin dans la séquence des codes ASCII.*
RetVal = Compare$(Str1, Str2, 1) *RetVal vaut alors 0 parce que, abstraction faite de la casse, les deux chaînes sont identiques.*
Str2 = Str1
RetVal = Compare$(Str1, Str2, 0) *RetVal vaut encore 0, puisque maintenant les deux chaînes sont identiques.*

FloatToString$(float) *Convertit un nombre en virgule flottante en une chaîne.*

float = Nombre en virgule flottante à convertir.

string = Variable chaîne devant contenir les données de la conversion.

string = FloatToString(float)

NumberToHex$(number) *Convertit un nombre en une chaîne hexadécimale.*

number = octet ou nombre entier à convertir.

string = Variable chaîne devant contenir les données de la conversion.

string = NumberToHex$(number)

StringToInt$(string) *Convertit une chaîne de données ASCII numériques en une valeur de donnée numérique entière.*

string = Variable chaîne contenant les données ASCII numériques.

Retourne les données numériques de la chaîne au format décimal.

number = StringToInt$(string)

StringToFloat$(string) *Convertit une chaîne de données ASCII numériques en une variable en virgule flottante.*

string = Variable chaîne contenant les données ASCII numériques.

Retourne les données numériques de la chaîne au format de virgule flottante.

float = StringToFloat$(string)

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. Quand une interruption survient, le processeur quitte le programme en cours d'exécution et exécute la macro associée à l'interruption puis il reprend le programme normal là où il l'avait quitté. La macro d'interruption doit être créée par le concepteur qui a implanté cette interruption.

Le nombre et le type d'interruptions disponibles dépendent du microcontrôleur utilisé. Certains processeurs disposent de nombreuses interruptions, d'autres n'en ont que peu.

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. Cependant Flowcode se sert de quatre type principaux :

TMR<X> - overflow : réagit à une fin de temporisation liée à un "timer" interne du processeur.

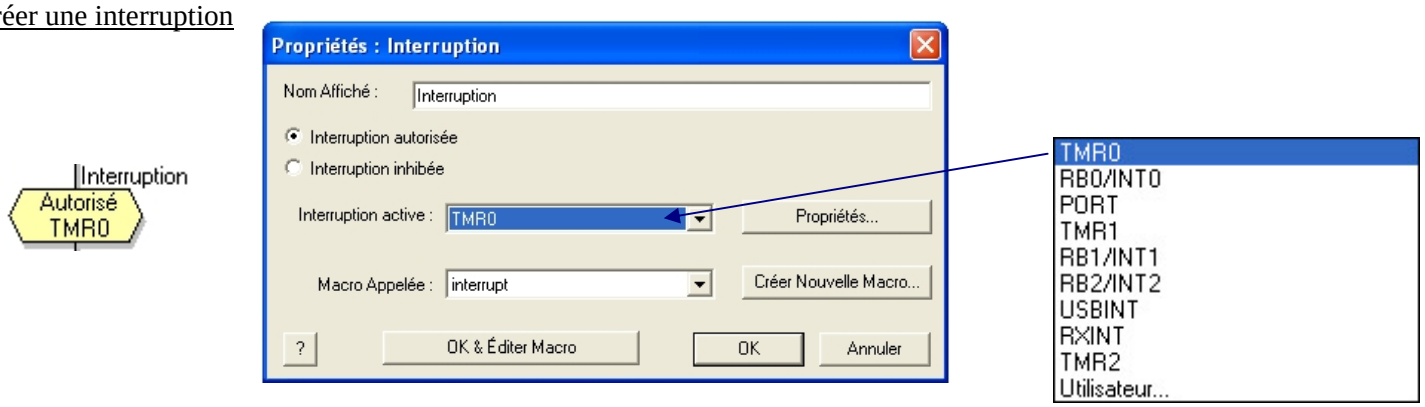
INT : réagit à un changement d'état logique sur une broche du processeur configurée en interruption externe.

Port change : réagit à un changement d'état logique sur un ensemble de broches du processeur.

Défini par le client : c'est le concepteur qui détermine la procédure d'interruption

Vous trouverez ci-après des précisions sur les différentes interruptions, accompagnées d'exemples.

Créer une interruption



On insère la fonction dans l'algorigramme. Ensuite, on valide ou non l'interruption pour l'utiliser ou la masquer. Sélectionnez la source d'interruption désirée dans le menu déroulant "Interruption active" parmi celles disponibles sur le processeur utilisé. Spécifiez les propriétés pour cet événement.
Choisissez la macro à appeler ou créez-en une nouvelle.

Propriétés des interruptions :

TMR<X> Interruption par dépassement de capacité d'un temporisateur (par ex. TMR0 overflow)

Arrête l'exécution du programme normal et exécute la macro d'interruption à chaque fois que le temporisateur spécifié arrive à bout de course. La temporisation s'effectue par comptage des impulsions de l'horloge avec application d'un facteur de division (une explication suit). Quand le compteur atteint le maximum, il retombe à zéro en provoquant un dépassement de capacité (overflow) qui lance la procédure d'interruption. Les interruptions par débordement de temporisateur se répètent périodiquement, ce qui se révèle très pratique pour déclencher une procédure à intervalle fixe ou pour des actions répétitives fréquentes, comme la mise à jour d'un afficheur.

Nota : Vérifiez que la vitesse d'horloge est correcte (boite de dialogue "Project Options" du menu "Edition") parce qu'elle affecte le réglage de la fréquence d'interruption du temporisateur.

Ce sont le processeur et son temporisateur qui déterminent les propriétés exactes disponibles. Aussi les réglages sur le dépassement du temporisateur peuvent-ils varier d'un processeur à l'autre et même d'un temporisateur à l'autre. L'exemple ci-dessous donnera une idée des caractéristiques que l'on peut rencontrer avec une interruption par débordement.

Des informations complémentaires sont données dans les boîtes de dialogue d'interruption, mais aussi dans les notes techniques des processeurs.
Par exemple TMR1 sur le 16F877A a moins d'options de pré-diviseur que l'interruption TMR0, il ne permet pas non plus le choix du flanc de déclenchement comme sur TMR0.

Exemple de propriétés de TMR0 (Timer0) :

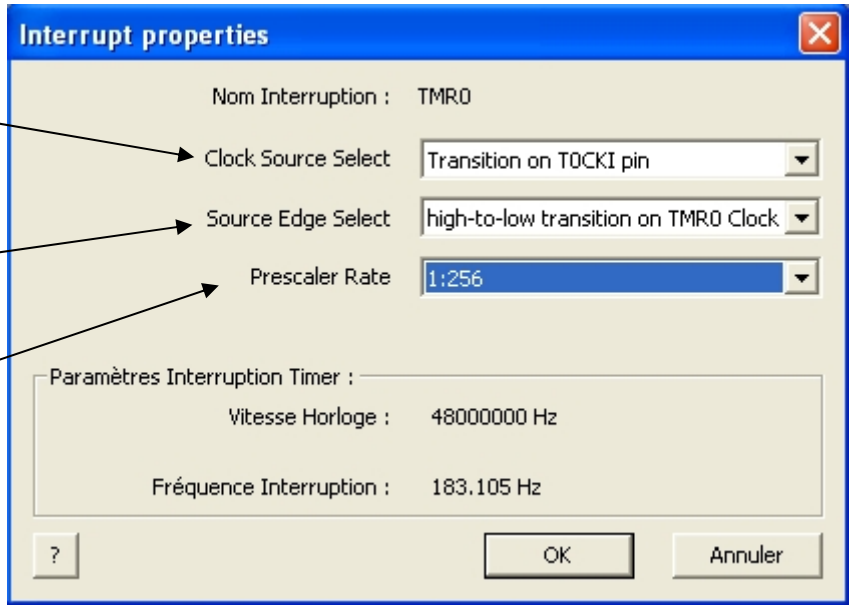
- Sélection de l'horloge de référence

Dans la sélection de la référence temporelle prise en considération pour la temporisation, les options dépendent du processeur utilisé.
Exemples :

- Horloge interne (CLK0)
- Transition sur T0CKI (broche "RA4" pour ECIO)

- Sélection du flanc de transition sur T0CKI pris en compte
  - Flanc montant (low-to-high transition)
  - Flanc descendant (high-to-low transition)

- Rapport du pré-diviseur : Sélection d'un rapport de division du signal d'horloge.



Le pré-diviseur réduit la fréquence de l'horloge utilisée pour le déclenchement du temporisateur, ce qui diminue la récurrence de l'exécution de la procédure d'interruption. Ce rapport peut prendre un certain nombre de valeurs différentes, selon le processeur choisi.

Par exemple : pour le Timer0 du PIC18F4455 utilisé dans le circuit ECIO40, le rapport de prédivision est compris entre 1/1 et 1/256 et Flowcode utilise ce timer en 8 bits.

Formule : fréquence d'interruption = (Fhorloge/4)*Rapport du prédiviseur/$2^{\text{nbre bits timer}}$ (Fhorloge/4 = fréquence d'instruction interne).

Fréquence d'horloge : 48 000 000Hz (48MHz)
Rapport du pré-diviseur : 1 / 256
Fréquence d'interruption : 183,105Hz

Fréquence d'horloge : 48 000 000Hz (48MHz)
Rapport du pré-diviseur : 1 / 64
Fréquence d'interruption : 732,422Hz

Nota : avec les ECIO, Flowcode V4.2 permet les interruptions par le Timer0 (mode 8 bits), le Timer1 (mode 16 bits) et le Timer2 (mode 8 bits), toutes les possibilités ne sont donc pas offertes.

INT

Procédure d'interruption déclenchée par le changement d'état logique sur une broche d'entrée du processeur configurée en "Interruption externe".

On peut choisir de déclencher l'interruption sur :

- le flanc descendant de INT
- le flanc montant de INT

Les interruptions INT sont utiles dans le cas où l'on souhaite que l'interruption s'exécute quand un événement externe particulier se produit, comme une commande d'arrêt d'urgence par exemple.

Nota : sur le PIC18F4455 (ECIO40) les broches RB0, RB1, RB2 permettent l'interruption INT (INT0, INT1, INT2)

Changement sur le Port

La procédure d'interruption s'exécute lors d'un changement d'état logique sur un port précis du microcontrôleur.
Nota : sur le PIC18F4455 (ECIO40) cette option s'applique au port B.

Client

L'option Client permet la création d'une procédure d'interruption entièrement définie par le concepteur. Cette option est utile pour exploiter des sources d'interruption présentes sur certains microcontrôleurs et non implémentées directement par Flowcode (USART, comparateur analogique…)

Propriétés de l'icône CODE C 

Des programmes écrits en C (ou/et en Assembleur) peuvent être "enfouis" dans une application Flowcode grâce à l'icône Code C.

Remarque : 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'algorigramme

Code C

Entrer le code C que vous souhaitez inclure à votre algorigramme. Le code C n'est pas contrôlé par Flowcode mais est transmis directement au compilateur C lorsque l'algorigramme 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 algorigramme.

- 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'algorigramme principal de Flowcode doivent contenir le préfixe FCC_Main_.
- Pour entrer un caractère Tab dans la fenêtre du Code C, utiliser Ctrl+Tab.

Code assembleur

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
}
```

- Pour accéder aux variables Flowcode ainsi qu'aux macros et aux points de jonction, il est nécessaire de caractériser l'élément utilisé par un préfixe précédé du caractère _ (souligné), à savoir _FCV_, _FCM_ et _FCC_NomMacro_ respectivement.

  Les exemples précédents deviennent alors _FCV_TEMPO, _FCM_TEST() et _FCC_TEST_A.

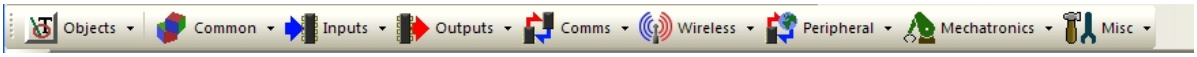
Propriétés de l'icône COMMENTAIRE 

L'icône Commentaires vous permet de placer des commentaires sur votre algorigramme.
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.

Remarquez que le commentaire réside sur le côté de l'algorigramme. En effet, il ne doit pas être exécuté et ne fait donc pas partie du programme en tant que tel.

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… Lorsqu'on travaille en équipe, un code bien documenté permet de mieux le comprendre et d'accélérer son développement.

La barre d'ICÔNES DE COMPOSANTS



Flowcode dispose de nombreux "composants" que l'on peut placer sur le panneau de simulation (panel).

Ils correspondent soit à des composants physiques externes utiles pour la simulation, tels que des boutons poussoirs ou commutateurs, des afficheurs divers (LEDs, 7 segments, LCD…), soit à des fonctionnalités internes particulières du microcontrôleur en liaison avec des circuits externes (convertisseur(s) A/N, commande(s) PWM, bus CAN, I²C, USB…).

Ces composants sont configurables par boîte de dialogue et peuvent nécessiter une ou plusieurs routines (sous-programmes) pour fonctionner. Toutes les routines sont fournies et ne demandent qu'un paramétrage limité. Pour certains composants simples, tels que LED ou commutateur, dans un souci de formation l'utilisation des routines fournies ne s'impose pas, il est donc préférable de gérer directement ces composants.

Cette notice se limite à décrire la mise en œuvre de quelques composants classiques.


Propriétés du composant SWITCH (interrupteur) ou SWITCHbank (rangée d'interrupteurs)



Propriétés du composant LED (LED seule) ou LEDarray (matrice de LEDs)



Propriétés du composant led7seg (simple afficheur 7 segments)

Propriétés du composant led7seg4 (quadruple afficheur 7 segments)


Propriétés du composant LCDDisplay (afficheur LCD)


Propriétés du composant ADC (convertisseur Analogique/Numérique)

Propriétés du composant PWM (modulateur MLI)

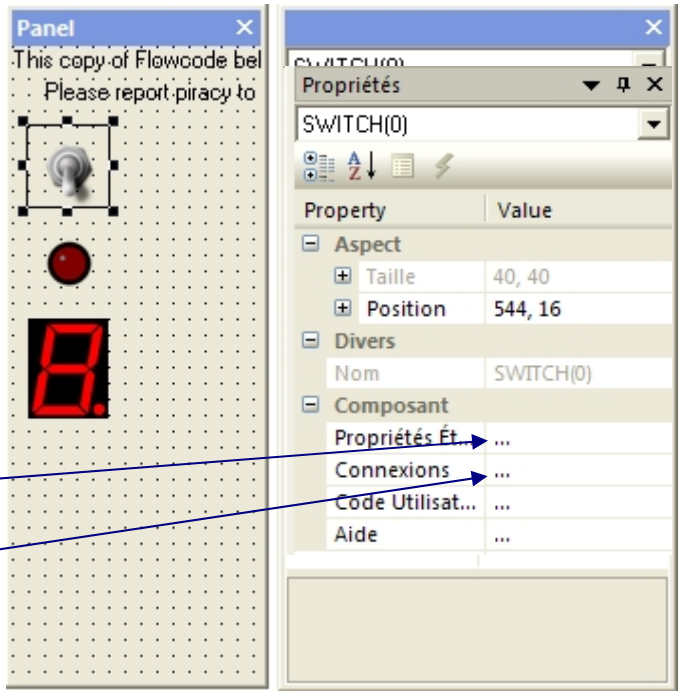

Propriétés du composant Stepper (moteur pas à pas)

Le PANNEAU de simulation et les COMPOSANTS

Lors d'un clic sur l'icône du composant dans la barre d'outil, celui-ci est posé sur le panneau de simulation (panel).
Nota : Si ce panneau n'est pas visible, ouvrir Affichage dans la barre de menu et cocher "Panel". Par commodité il est préférable de rendre ce panneau flottant (fenêtre Panel, clic droit sur Panel, sélection de "Floating") puis d'ajuster sa taille.

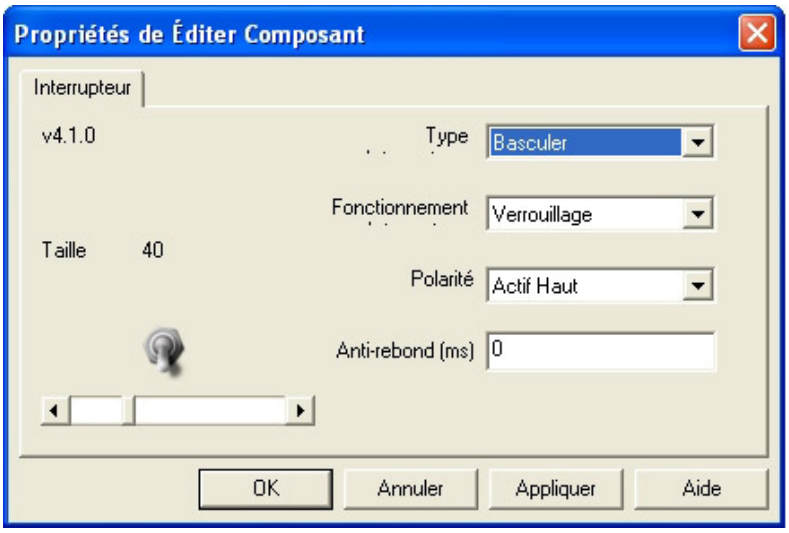
Dans la fenêtre de Propriétés liée au panneau on peut paramétrer chaque composant, en particulier :

- ses propriétés physiques (type, couleur, taille…) en cliquant à doite de Propiétés Étendues,
- ses connections au microcontrôleur en cliquant à doite de Connexions.



Propriétés du composant SWITCH (interrupteurs) 

➢ Propriétés Étendues



Taille :
Permet d'ajuster la taille du pictogramme affiché sur le panneau de simulation.

Type de commutateur :
Utiliser la liste déroulante pour sélectionner le type de commutateur .

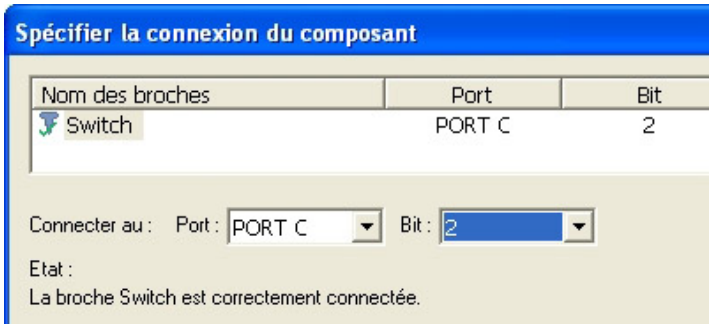
Types disponibles :

- à levier (basculer)
- Bouton-poussoir
- à glissière (inverseur)
- à bascule (rocker)
- Rotatif

Fonctionnement :
Configure si le contact a un fonctionnement bistable ("verrouillage") ou monostable ("momentané")

Type de contact : Polarité
Configure si le commutateur produit un 1 logique (Actif Haut) ou un 0 logique (Actif Bas) lorsqu'il est actionné.

Anti-Rebond :
Temps de rebond du contact en ms. Le temps de rebond est utilisé en relation avec les routines Délai_de_Montée et Délai_de_Descente pour s'assurer que les rebonds sont terminés et que l'état stable du contact est obtenu.

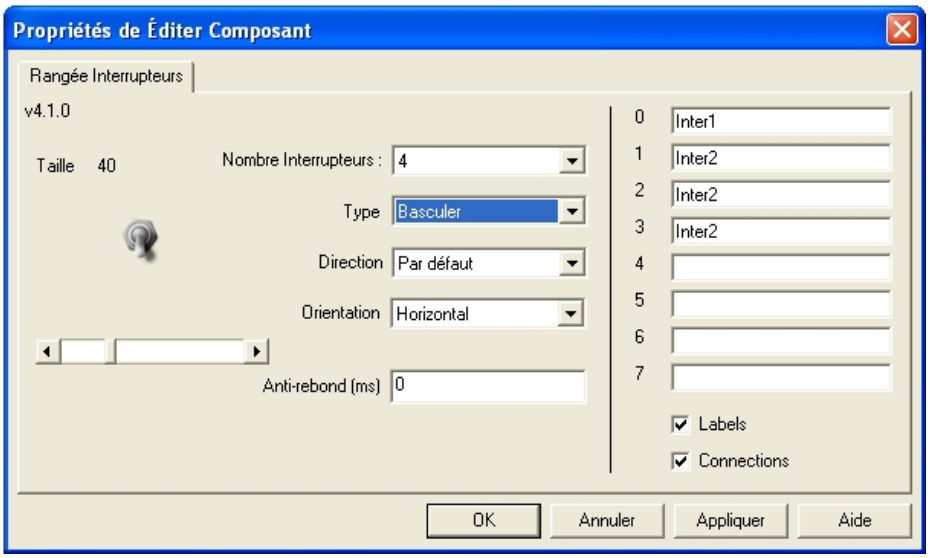


➢ Connexions

Préciser la ligne de port à laquelle est relié le commutateur.
La ligne Etat précise si la connexion est valide ou non.

Propriétés du composant SWITCHbank (rangée d'interrupteurs) 

➢ Propriétés Étendues



Nombre de commutateurs :
Donner une valeur entre 1 et 8

Type de commutateur :
Deux types possibles : à levier ou à poussoir (contact à fermeture)

Direction : ordre d'affichage des commut.

Orientation :
Disposition de la rangée (Horizontale ou verticale)

Taille et rebond : voir SWITCH ci-dessus

Cocher Labels et/ou Conections pour afficher les étiquettes et/ou les lignes de port sur le panneau.

➢ Connexions

Voir SWITCH ci-dessus

Remarque : il est généralement plus judicieux d'utiliser la rangée d'interrupteurs, même pour un seul, car ce composant permet de voir à quelles broches ils sont raccordés et de leur affecter des étiquettes.

Propriétés du composant LED

➢ Propriétés Étendues

Le composant LED présente les propriétés réglables suivantes :

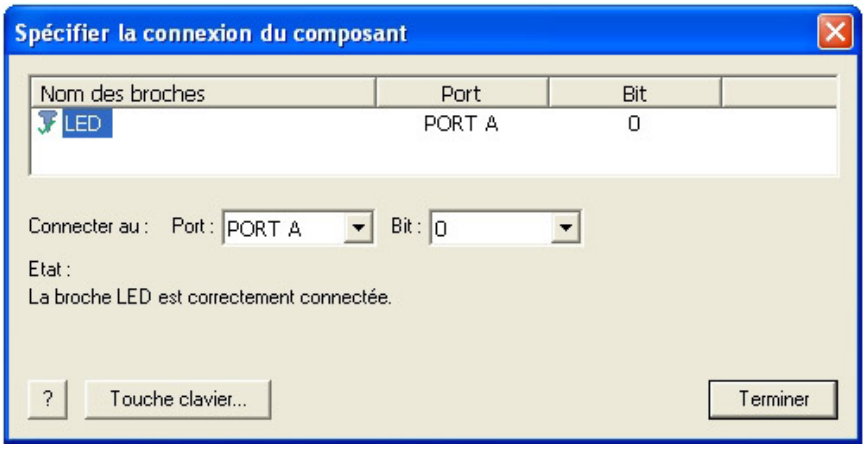
Taille :
Permet d'ajuster la taille du pictogramme affiché sur le panneau de simulation.

Forme :
Permet de définir la forme du pictogramme affiché sur le panneau de simulation.

Couleur :
Permet de définir la couleur du pictogramme affiché sur le panneau de simulation.

Polarité :
Permet de définir le niveau logique qui provoque l'allumage de la LED
Actif Haut : la LED s'allume avec un niveau logique 1
Actif Bas : la LED s'allume avec un niveau logique 0

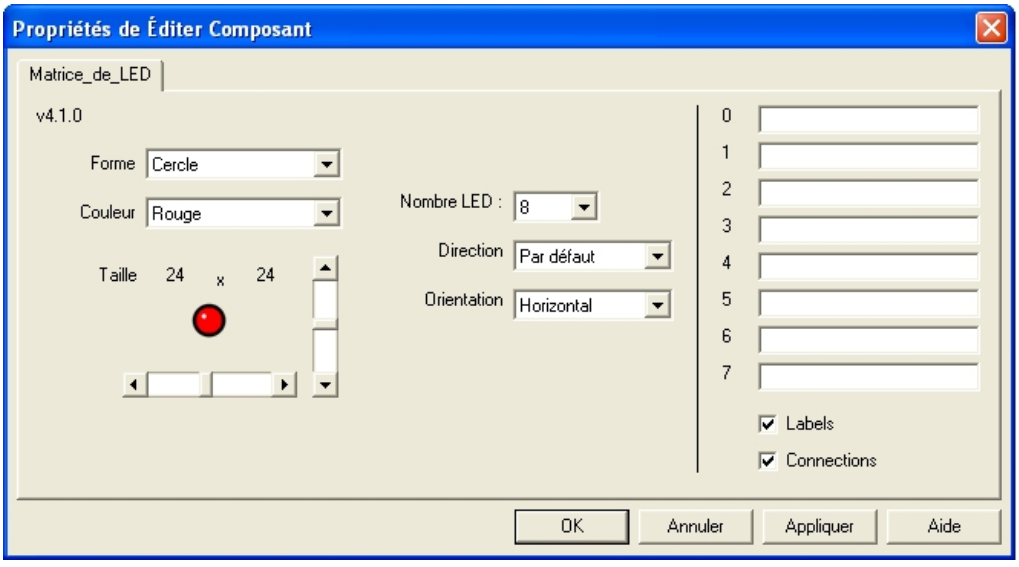
➢ Connexions



Préciser la ligne de port du microcontrôleur à laquelle est relié la LED.
La ligne Etat précise si la connexion est valide.

Propriétés du composant LEDarray (matrice de LEDs)

➢ Propriétés Étendues



Les propriétés du composant LEDarray sont identiques à celles du composant LED (voir ci-dessus).

On peut régler en plus le nombre de LED, l'orientation du groupement.

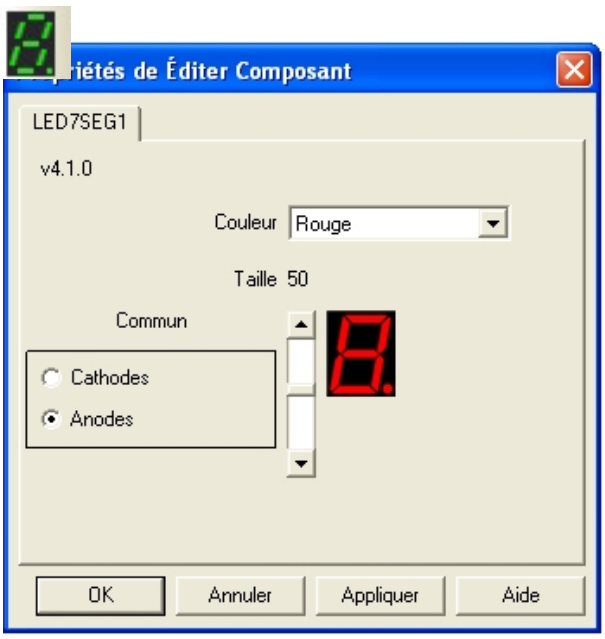
Cocher Labels et/ou Conections pour afficher les étiquettes et/ou les lignes de port sur le panneau.

➢ Connexions

Préciser le port du microcontrôleur auquel sont reliées les LEDs (toutes les LEDs du groupement doivent être connectées au même port).
La ligne Etat précise si la connexion est valide.

Remarque : il est généralement plus judicieux d'utiliser la matrice de LEDs, même pour une seule, car ce composant permet d'afficher à quelles broches elles sont raccordées et de leur affecter des étiquettes.

Propriétés du composant led7seg (simple afficheur 7 segments)



➢ Propriétés Étendues

Le composant LED présente les propriétés réglables suivantes :

Commun :
Lié au type d'afficheur utilisé : cathode commune ou anode commune.

Couleur :
Permet de définir la couleur du pictogramme affiché sur le panneau de simulation.

Taille :
Permet d'ajuster la taille du pictogramme affiché sur le panneau de simulation.

➢ Connexions

Préciser le port du microcontrôleur auquel est relié l'afficheur
Note :

- Tous les segments doivent être connectées au même port
- dans le cas où le point digital est utilisé, il faut relier le point commun des segments à une ligne d'un autre port.

La ligne Etat précise si la connexion est valide.

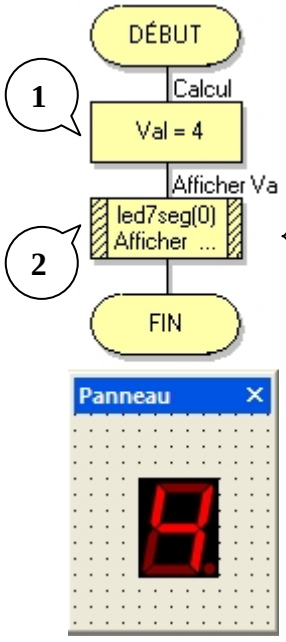
➢ Routines composant

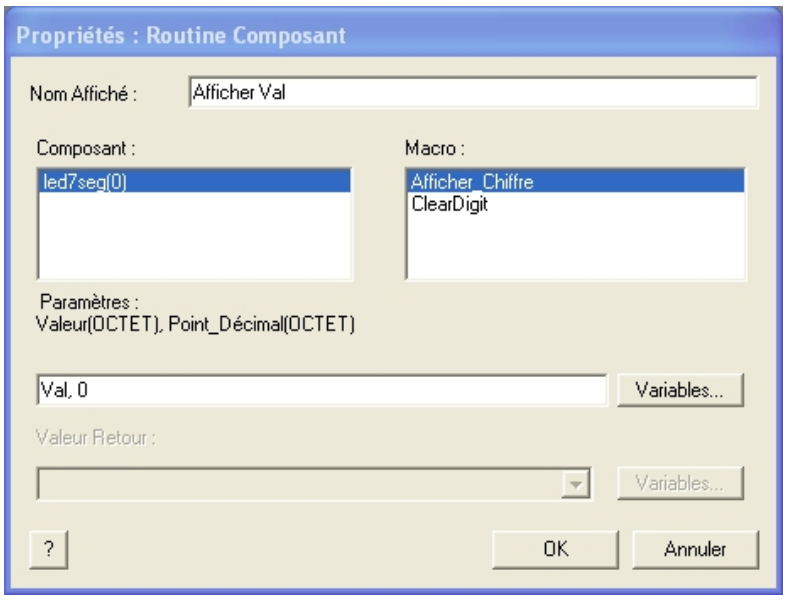
Deux routines composant sont préécrites et permettent donc d'utiliser très facilement ce composant :

- Afficher_chiffre : allume les segments en fonction de la valeur du paramètre Valeur et le point décimal en fonction de la valeur Point_ Décimal.

Dans l'exemple ci-contre on affecte la valeur 4 à la variable Val (1) que l'on fait afficher ensuite (2). Le point décimal est éteint.

- Cleardigit : éteint l'afficheur.



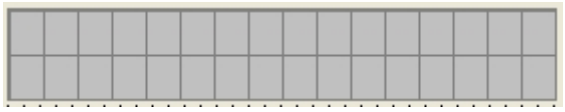


Propriétés du composant led7seg4 (quadruple afficheur 7 segments)

La mise en œuvre de ce composant est sensiblement identique à celle du simple afficheur 7 segments ci-dessus. Les différences se situent au niveau :

- Des connexions : il y a 4 broches communes au lieu d'une seule. L'affichage étant multiplexé les segments sont en parallèle. Suivant l'utilisation ou non des points décimaux cet afficheur nécessite donc entre 11 et 15 lignes de port.
- Des routines composant : il n'existe que la routine Afficher_chiffre qui comporte 3 paramètres :
  - Chiffre : spécifie l'afficheur à allumer (les afficheurs sont numérotés de 0 à 3 en partant de la gauche, une valeur de digit supérieure à 3 éteint les afficheurs)
  - Valeur : correspond à la valeur à afficher sur l'afficheur sélectionné.
  - Point_ Décimal : spécifie si le point décimal de l'afficheur sélectionné doit s'allumer ou non

Propriétés du composant LCDDisplay (afficheur LCD)



Nota : L'afficheur fourni en Eblocks (EB005) comporte **2 lignes de 16 caractères**

➢ Propriétés Étendues

Format LCD :
Définit le nombre de cractères par ligne x le nombre de lignes.

Taille Texte :
Permet de définir la hauteur des caractères (en pixels) du pictogramme affiché sur le panneau de simulation.

Couleur Texte:
Permet de définir la couleur des caractères sur le pictogramme affiché sur le panneau de simulation.

Couleur Arrière-plan:
Permet de définir la couleur de l'arrière-plandes caractères sur le pictogramme affiché sur le panneau de simulation.

➢ Connexions

Préciser le port du microcontrôleur auquel est relié l'afficheur
Nota : Les connexions indiquées par défaut correspondent à celles de l'Eblock EB005. Dans le cas de l'utilisation de ce circuit seule la lettre de port peut donc être changée.

La ligne Etat précise si la connexion est valide.

➢ Routines composant

Dix routines composant sont préécrites et permettent donc d'utiliser très facilement ce composant :

Init : Cette routine doit être appelée une fois pour initialiser l'afficheur LCD avant toute autre routine le concernant.

Effacer : Efface l'afficheur

Effacer_Ligne (Ligne (OCTET)) : Efface la ligne dont le numéro correspond à la valeur de **Ligne**

Écrit_Caractère (Caractère (OCTET)) : écrit le caractère correspondant au code ASCII de la valeur caractère.

Écrit_Nombre (Nombre (ENTIER)) : écrit le nombre.

Écrit_Chaîne (Chaine Caractères (CHAINE)) : écrit la chaîne de caractères.

Commande (in (OCTET)) : Envoie un octet de commande à l'afficheur (consulter la fiche détaillée de l'afficheur)

Curseur(x(OCTET), y(OCTET)) : Positionne le curseur dans une position donnée en **x** et en **y**

Dérouler_Affichage(Direction (OCTET), Nombre_Positions (OCTET)) : fait défiler l'affichage vers la gauche ou la droite en fonction du nombre de positions correspondant à la variable **Nombre_Positions**.

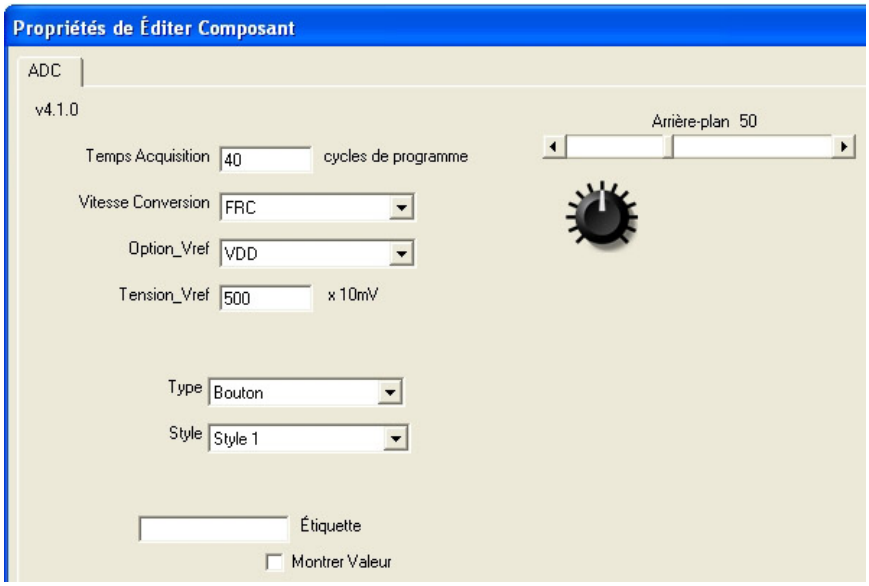
Si l'octet **Direction** vaut 0 l'affichage défile vers la gauche
Si l'octet **Direction** vaut 0 l'affichage défile vers la gauche

Écriture_en_RAM (nldx(OCTET), d0(OCTET)… d7(OCTET)) : modifie la mémoire interne de l'afficheur afin d'y intégrer jusqu'à 8 caractères personnalisés.

Les caractères sont numérotés de 0 à 7 grâce à la valeur de **nldx**.
Les octets d0 à d7 correspondent aux valeurs des données en colonne du caractère personnalisé.
Pour afficher un de ces caractères il faut utiliser la routine **Écrit_Caractère**
Nota : cette routine n'est pas simulable.

Propriétés du "composant" ADC (convertisseur Analogique/Numérique)

Nota : il ne s'agit pas exactement d'un composant mais d'une fonction interne particulière des microcontrôleurs. Ce composant apparaît sur le panneau de simulation sous la forme d'un potentiomètre, rectiligne ou rotatif, animable à la main.



➢ Propriétés Étendues

Temps Acquisition :

Définit la durée de charge du condensateur de l'échantillonneur/bloqueur avant de lancer la conversion.
Si un seul canal de conversion Analogique/Numérique est utilisé le réglage peut être 0 ou 1 pour maximiser la vitesse de conversion, sinon la valeur par défaut de 40 est recommandée.

Vitesse Conversion :

Définit la durée de conversion $T_{A/D}$.
Si choix FRC => $T_{A/D}$ = 1µs
Sinon formule : $T_{A/D}$ = 1/VitesseConversion
Ex : VitesseConversion = FOSC/16
Si FOSC = 48MHz (ECIO40) => $T_{A/D} = (16/48)*10^6 = 0{,}33\mu s$
Nota : sachant que $T_{A/D}$ mini = 0,8ms, si FOSC = 48MHz seules les vitesses de conversion FRC et FOSC/64 conviennent.

Option_Vref :

Définit la source pour la tension de référence du convertisseur.
- VDD : la référence est fournie par le +5V du microcontrôleur.
- VREF+ : la référence est fournie par une source annexe (conversion plus précise qu'avec VDD)

Tension_Vref :

Définit la valeur de la tension de référence du convertisseur lorsque l'option VREF+ est sélectionnée.

Type :

Le convertisseur ADC est matérialisé sur le panneau de simulation par le pictogramme d'un potentiomètre animable manuellement : soit potentiomètre tournant (bouton), soit potentiomètre rectiligne (curseur).

➢ Connexions

Préciser la ligne de port du microcontrôleur utlisée en entrée ADC (lié au microcontrôleur utilisé).
La ligne Etat précise si la connexion est valide.

➢ Routines composant

Quatre routines composant sont préécrites et permettent donc d'utiliser très facilement ce composant :

Lire_comme_Octet : Renvoie les 8 bits de poids fort de la valeur analogique convertie.
Exemple : avec un convertisseur de résolution >=8 bits, la valeur renvoyée est comprise entre 0 et 255 par incrément de 1

Lire_comme_Entier : Renvoie la valeur analogique convertie sous forme d'Entier (nombre à 16 bits)
Nota : tous les bits ne sont pas significatifs. Il faut se référer à la documentation du microcontrôleur pour connaître la résolution du (des) convertisseur(s) ADC.
Par exemple, sur le PIC 18F4455 utilisé dans l'ECIO40, la résolution est de 10 bits. La valeur renvoyée est donc comprise entre 0 et 1023 par incrément de 1.

Lire_comme_Tension : Renvoie la valeur analogique de la tension convertie sous forme de nombre à virgule flottante (nombre à 32 bits)
Exemple : avec VDD = 5V et un convertisseur 10 bits la valeur renvoyée est comprise entre 0.00 et 5.00 par incrément de 0.05

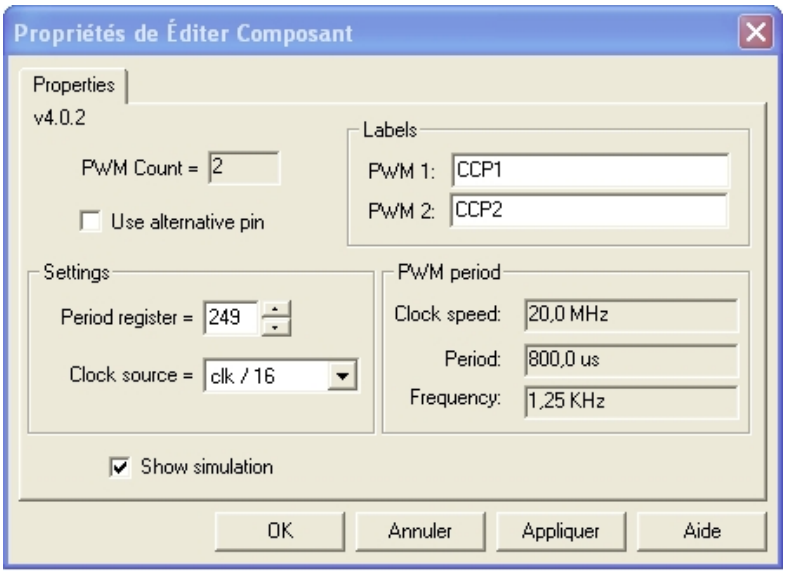
Lire_comme_Chaîne : Renvoie la valeur analogique de la tension, convertie en nombre à virgule flottante, sous forme de chaîne de caractères (intéressant en association avec un afficheur LCD).

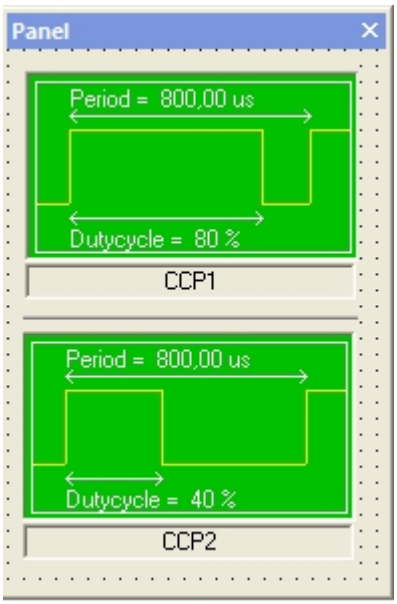
Propriétés du "composant" PWM (Pulse Width Modulation = MLI modulateur de largeur d'impulsion) (sous-menu Mechatronics)

Nota : il ne s'agit pas exactement d'un composant mais d'une fonction particulière de certains microcontrôleurs. Lors de la simulation on peut observer les graphes des signaux MLI sur le panneau de simulation.

➢ Propriétés Étendues

Les copies d'écran ci-dessous correspondent à un microcontrôleur PIC 18F4455 possédant deux canaux PWM, CCP1 et CCP2, correspondant respectivement aux broches 17 et 16 (ou 36 si on utilise la broche alternative (alternative pin)).





Period register :
Nombre de valeurs possibles du rapport cyclique = Period register +1.

Clock source :
Valeur du prédiviseur de la fréquence d'horloge du microcontrôleur.

En pratique il faut combiner les 2 réglages pour obtenir la fréquence de modulation (Frequency) souhaitée en fonction de la fréquence d'horloge du microcontrôleur (Clock speed).

➢ Connexions

Pas de connexions à spécifier pour ce composant.

➢ Routines composant

Cinq routines composant sont préécrites et permettent donc d'utiliser très facilement ce composant :

Enable (nldx(OCTET)) : Valide le canal PWM spécifié et lance son fonctionnement
Dans l'exemple ci-dessus la routine est appelée 2 fois : une 1ère fois avec nldx = 1 pour activer CCP1 et une 2ème fois avec nldx = 2 pour activer CCP2.

Disable (nldx(OCTET)) : Désactive le canal PWM spécifié.

SetDutyCycle (nldx(OCTET), nDuty(OCTET)) : Fixe la valeur du rapport cyclique (nDuty) du canal PWM spécifié (nldx).
Dans l'exemple ci-dessus la routine est appelée 2 fois :
- une 1ère fois pour CCP1 (nldx = 1) avec nDuty = 200 => le rapport cyclique = 200/250 = 80%
- une 2ème fois pour CCP2 (nldx = 2) avec nDuty = 100 => le rapport cyclique = 100/250 = 40%

SetDutyCycle10bit (nldx(OCTET), nDuty(ENTIER)) : Fixe la valeur sur 10 bits du rapport cyclique (nDuty) du canal PWM spécifié (nldx).
Le rapport cyclique varie de 0% (nDuty = 0) à 100% (nDuty = 1023)

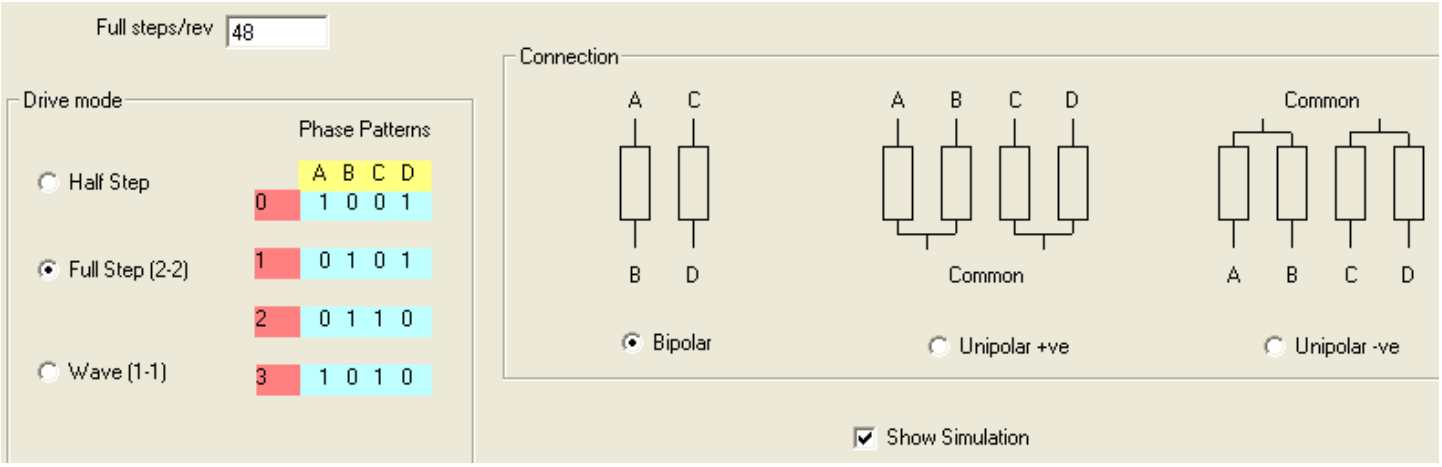
ChangePeriod (nPeriodVal(OCTET), nPrescalerVal(ENTIER)) : Permet de modifier en cours d'exécution les valeurs de Period register et de Clock source (à utiliser avec précaution).

Propriétés du composant STEPPER (Moteur pas à pas, sous-menu Mechatronics)

Ce composant permet la mise en oeuvre de moteurs pas à pas en utilisant des routines intégrées dans Flowcode. Il permet de gérer directement plusieurs types de moteurs pas à pas (unipolaire ou bipolaire) et aussi plusieurs modes de fonctionnement de ces moteurs.

Nota : les moteurs pas à pas ne peuvent pas être commandés directement par les broches du microcontrôleur, il est donc nécessaire d'insérer un circuit d'interface de puissance. L'algorigramme peut donc avoir à prendre en compte certaines spécificités de cette interface.

- **Propriétés Étendues**



Pas entiers par tour (*Full steps/rev*) :
Définit le nombre de pas nécessaires pour que le moteur effectue un tour complet.
La valeur par défaut 48 correspond à un angle de pas de 7,5 degrés (360 / 7,5 = 48).

Mode de commande (*Drive Mode*) :
Tois modes sont utilisables :

- Pas entier 2 bobines en simultané (*Full step*) : permet d'exploiter le moteur avec son couple maximum.
- Pas entier 1 bobine (*Wave*) : alimentation d'une bobine à la fois. Peut être utilisé pour les applications de puissance inférieure.
- Demi-pas (*Half Step*) : double le nombre de pas par tour en alternant entre « Pas entier 2 bobines » et « Pas entier 1 bobine ».

Tableau de séquence (*Phase Patterns*) : montre la séquence des états des broches de sortie du microcontrôleur pilotant les bobinages du moteur.

Connexions :
Permet de spécifier le type de moteur pas à pas contrôlé via le composant « Moteur pas à pas ».

- Bipolaire (*Bipolar*) : ce type de moteur comporte 2 bobines. Cela signifie que pour contrôler correctement le moteur, il faut inverser le courant dans les bobines donc les tensions. Pour ce faire on utilise généralement une alimentation simple associée à un circuit d'interface de puissance avec double pont en H.
- Unipolaire (*Unipolar*) : ce type de moteur comporte 2 bobines à point milieu donc 4 demi-bobines. Le point milieu des bobines (common) est relié soit au potentiel + de l'alimentation (cas *unipolar –ve*), soit au potentiel - de l'alimentation (cas *unipolar +ve*)

Afficher la simulation (Simulation Show) : permet d'afficher l'animation du composant dans le panneau.

- **Connexions :**

Spécifier l'affectation des 4 extrémités de bobines du moteur (A, B, C, D) aux broches de pilotage du microcontrôleur via le circuit d'interface.

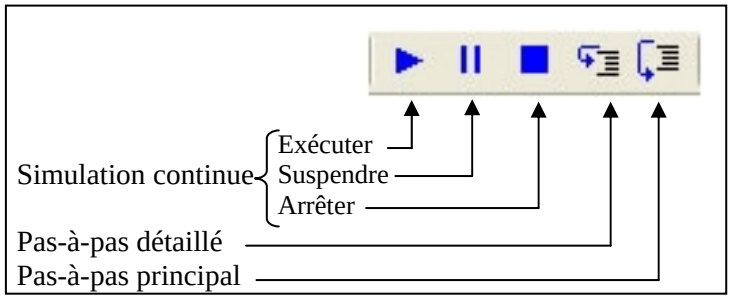
- **Routines composant :**

Quatre routines composant sont préécrites et permettent donc d'utiliser très facilement ce composant :

IncrementStep() : Incrémente d'un pas la position courante du moteur.
DecrementStep() : Décrémente d'un pas la position courante du moteur (inversion du sens).
EnableMotor() : Autorise le fonctionnement du moteur.
DisableMotor() : Désactive le fonctionnement du moteur (évite de faire chauffer le moteur avec le rotor fixe)

La simulation

Flowcode dispose de 3 modes de simulation :

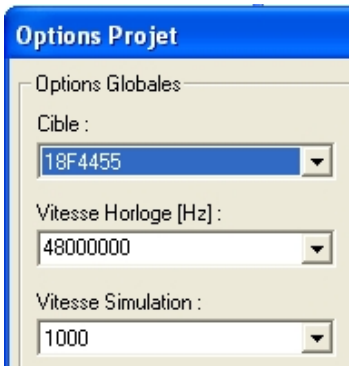


- **La simulation en continu** : correspond sensiblement au fonctionnement attendu du microcontrôleur in-situ.
- **Le pas-à-pas détaillé** : l'algorigramme est exécuté icône par icône à chaque clic de souris sur le bouton.
  - Un rectangle rouge signale l'icône en cours d'exécution.
  - Toutes les macros traitées en algorigrammes sont simulées.
  - Les fenêtres des variables, de la pile des appels de macros et du PICmicro sont mises à jour à chaque étape de la simulation.
- **Le pas-à-pas principal** : fonctionne comme le Pas à Pas détaillé, à la différence suivante : quand le mode Pas à Pas principal rencontre une macro contenant un algorigramme séparé, Pas à Pas principal traite la macro en entier au lieu de l'ouvrir et de l'exécuter pas à pas comme le ferait Pas à Pas détaillé.

Vitesse de simulation :

En mode "simulation en continu" la vitesse de simulation est réglable dans la boite de dialogue "Options Projet" du menu "Edition" afin d'observer le déroulement du programme.

Nota : Si le réglage "Aussi vite que possible" est sélectionné alors les fenêtres des variables, de la pile des appels de macros et les vues du microcontrôleur ne sont pas rafraîchies à moins de suspendre le déroulement de la simulation.



Les points d'arrêt :

- **Ajouter et enlever des points d'arrêts**

Les points d'arrêts peuvent être mis ou enlevés en sélectionnant l'icône sur laquelle vous voulez vous arrêter, puis en choisissant l'option Mettre/enlever un point d'arrêt du menu Edition. Une autre façon de faire est d'utiliser la touche fonction F9.
Pour effacer tous les points d'arrêt, sélectionner Effacer tous les points d'arrêt depuis le menu Edition.
Les points d'arrêt sont matérialisés par un point rouge en haut et à gauche de l'icône sélectionnée.

- **Utilisation de points d'arrêts**

Lorsque qu'un algorigramme est simulé en continu, la simulation s'arrête à la rencontre le premier point d'arrêt. Appuyer sur la touche Exécuter pour relancer le déroulement de la simulation jusqu'à la rencontre du prochain point d'arrêt s'il y en a ou jusqu'à la fin s'il n'y en a plus d'autres.
Lorsque la simulation rencontre un point d'arrêt, l'utilisateur peut examiner la valeur des variables, les entrées/sorties, les composants du panneau de simulation...
Les points d'arrêts sont bien pratiques pour arrêter le programme au début d'un morceau de code particulièrement complexe. L'utilisateur peut alors prendre la main pour avancer pas à pas et mettre au point son programme.

La compilation et le transfert d'un programme vers un PICmicro

Si on utilise les cartes de développement ECIO28 ou ECIO40 de Matrix Multimedia, toutes les options de compilation sont pré-renseignées, le mode opératoire est donc des plus simples.

Introduction
Les microcontrôleurs PICmicro peuvent seulement exécuter du code hexadécimal (format ;hex). Flowcode doit donc traduire les algorigrammes (format .fcf) en un format compréhensible par un composant PICmicro.

Flowcode procède de la façon suivante :

1 Traduit l'algorigramme en code C (format .c)
2 Compile le code C en Assembleur (format .asm)
3 Assemble le programme assembleur en code Hexa (format .hex)
4 Envoie le code Hexa au composant PICmicro

Bien que ceci puisse paraître un peu lourd, cette façon de faire procure l'avantage de permettre d'enfouir dans le programme du code C ou de l'assembleur provenant de tiers. Cela permet également au programmeur confirmé d'observer le code C ou le code assembleur générés pour cerner certains dysfonctionnements.

Nota : Avant de compiler l'algorigramme en un programme qui sera envoyé au microcontrôleur PICmicro, il faut vérifier que le choix de PICmicro cible est correct (boite de dialogue "Options Projet" du menu "Edition")

Compiler un programme vers une carte de développement ECIO28 ou ECIO40 :

- Vérifier le choix de la cible
- Raccorder la carte ECIO à un port USB du PC
- Lancer la compilation/transfert du programme vers le microcontrôleur (menu Puce Compiler -> Puce)

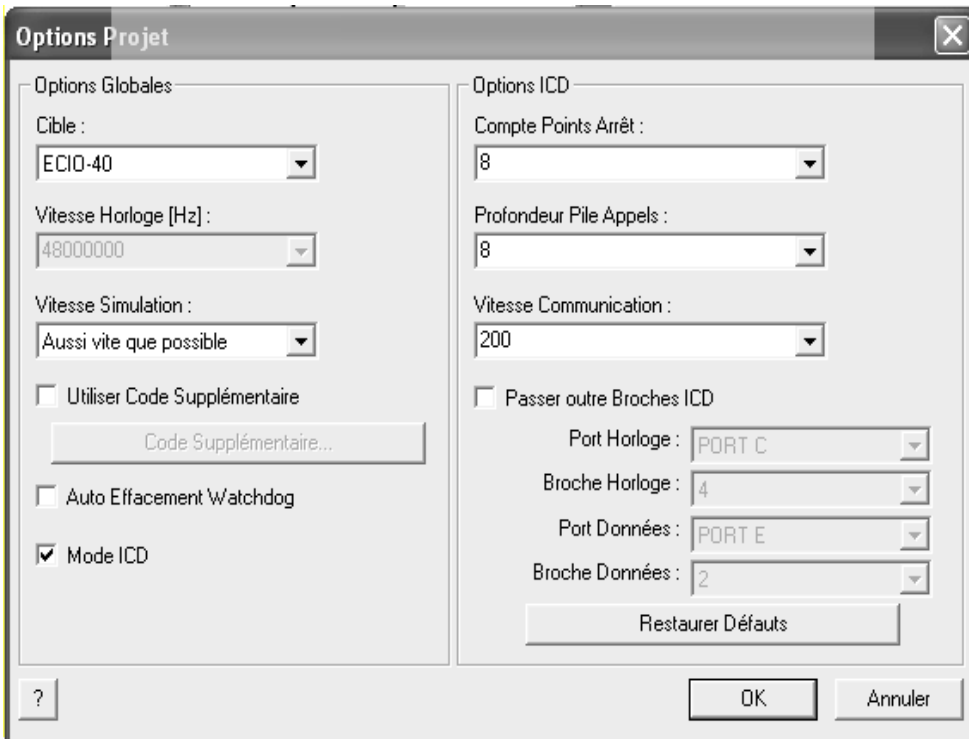
En fin de processus, une fenêtre avec le message "Please connect the ECIO…" s'affiche, il faut alors actionner le bouton Reset de la carte ECIO et le programme est transféré en mémoire du PICmicro.

Le débogage in-situ avec la carte FLOWKIT

Installer le driver pour Flowkit

Sous Flowcode : Options projet :

- ***ECIO40***
- Cocher ***Mode ICD***



Après la saisie de l'algorigramme :

- Raccorder l'ECIO au PC via la carte Flowkit, placer le commutateur de la carte Flowkit en position ***PROG***
- Création/Transfert du programme PIC :
  - Menu ***Puce*** : ***Compiler vers puce***. Au message "***Please connect the ECIO***…" appuyer sur le ***BP reset*** de la carte ECIO.
- Essais :
  - Placer le commutateur de la carte flowkit en position ***ICD***
  - Sous Flowcode les différents modes de simulation sont utilisables

  Attention : en mode Simulation continue il n'y a pas de mise en surbrillance de l'algorigramme sauf si on place des points d'arrêt.