Lycée Jules Ferry Versailles	**Comportement Séquentiel et Algorithmie** **Initiation au Microcontrôleur (FLOWCODE)**	S-SI
S si		***TP***

1. Présentation du Microcontrôleur

1.1. Objectif de la séance :

Cette séance permettra de comprendre le fonctionnement d'un microcontrôleur (ports, entrées, sorties), de concevoir et de tester son programme de manière graphique ainsi que d'implémenter le programme dans la mémoire et de réaliser les tests de fonctionnement.

1.2. Mise en situation :

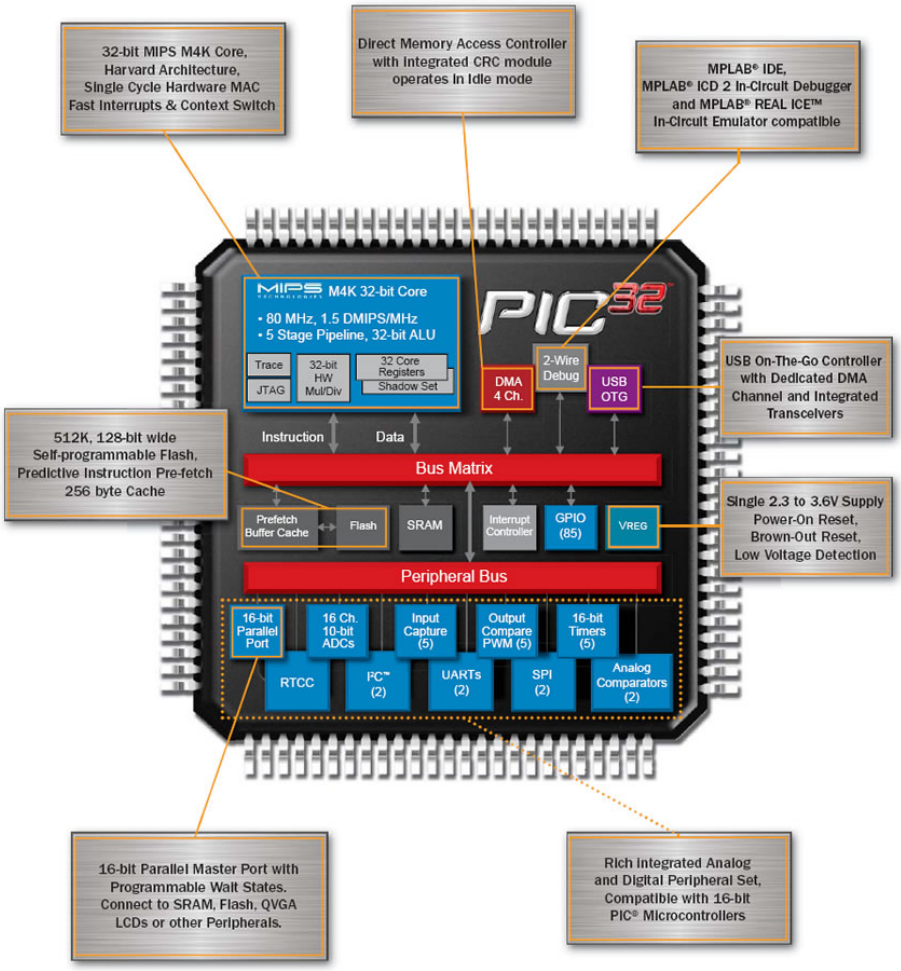
1.2.1. Le microcontrôleur :

Un **microcontrôleur** est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur :

- processeur,
- mémoires :
  - ROM : mémoire morte pour le programme,
  - RAM : mémoire vive pour les données),
- unités périphériques,
- interfaces d'entrées-sorties.

Les microcontrôleurs se caractérisent par rapport aux **microprocesseurs polyvalents** utilisés dans les ordinateurs personnels par :

- un plus haut degré d'intégration,
- une plus faible consommation électrique,
- une vitesse de fonctionnement plus faible (quelques MHz à quelques centaines de MHz),
- un coût réduit.

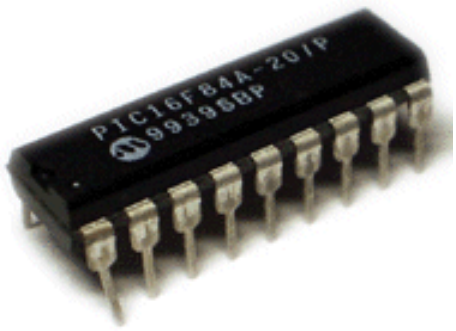


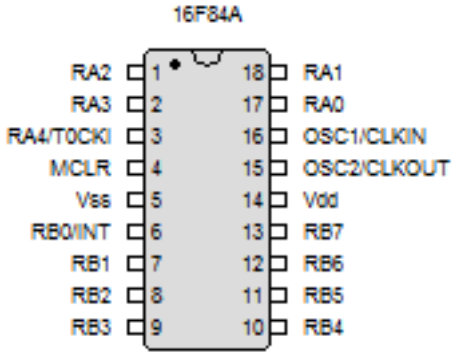
Le microcontrôleur est donc un circuit programmable, capable de gérer des périphériques, des entrées et des sorties logiques, numériques et analogiques.

Les **microcontrôleurs PIC** (ou **PICmicro** dans la terminologie du fabricant) forment une famille de microcontrôleurs de la société *Microchip*.

Le nom PIC n'est pas officiellement un acronyme, bien que la traduction en « Peripheral Interface Controller » (contrôleur d'interface périphérique) soit généralement admise.

1.3. Le microcontrôleur PIC 16F84A :



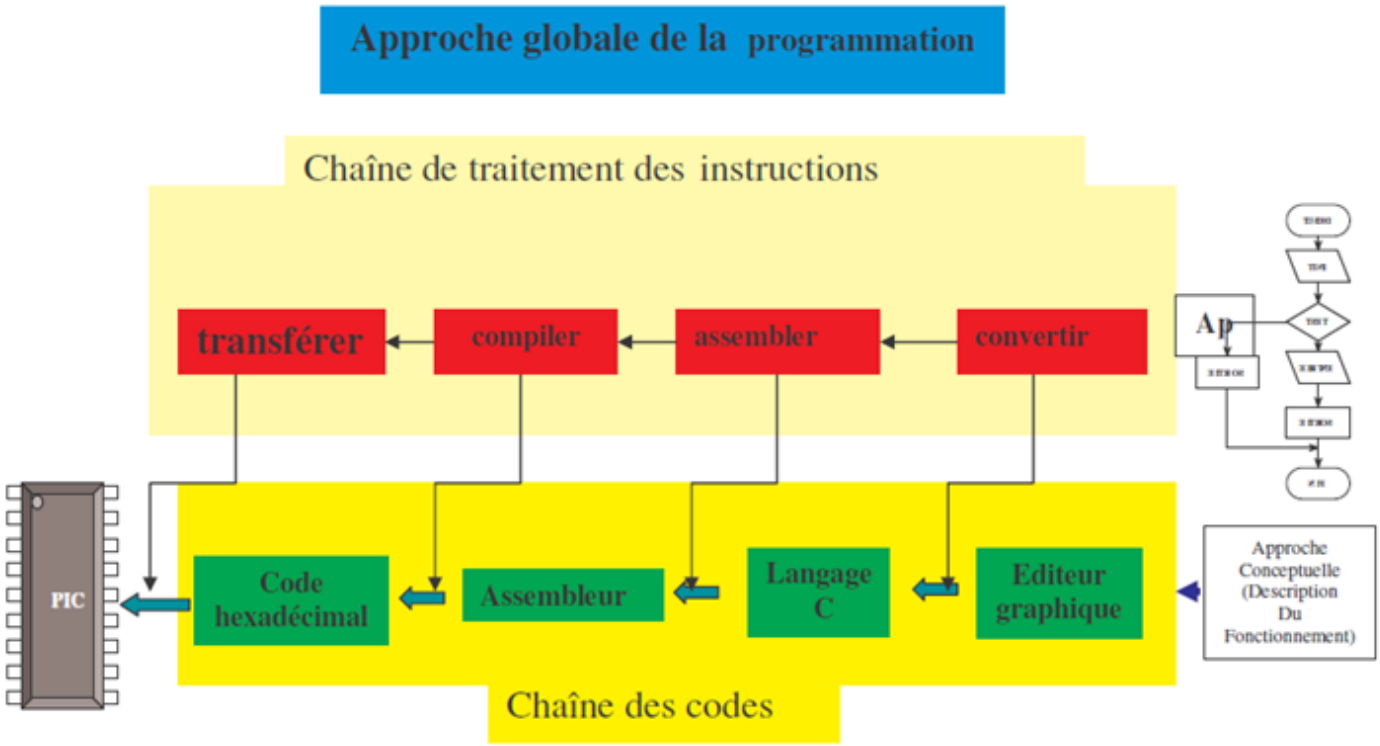


Le 16F84A, possède 18 broches.
Les broches 14 et 5 sont les broches d'alimentation du µC, respectivement **Vcc** (+5V) et **Vss** (0V).
La broche 4, **/MCLR** correspond au reset global du circuit. Il est actif au niveau bas.
Les broches 15 et 16 correspondent à l'horloge de fonctionnement du circuit. Cette horloge est réalisée soit avec un oscillateur (quartz) soit à l'aide d'une horloge externe. **La valeur de la fréquence de l'horloge devra être indiquée lors de la programmation du circuit.**
Le **PORTB**, comprend un octet (BYTE) de donnée (DATA) de RB0 à RB7. Chaque bit de cet octet peut être utilisé et configuré comme une entrée (I) ou comme une sortie (O), il est bidirectionnel. Le bit RB0 peut de plus, être utilisé comme un signal d'interruption (INT) externe.
Le **PORTA** est un octet tronqué, car il ne comprend que les 5 premiers bits de l'octet de RA0 à RA4. Chaque bit de ce PORT peut également être configuré en I/O. Le bit RA4 peut de plus être utilisé en qualité de timer0 (temporisateur/compteur)

1.4. Le logiciel FLOWCODE :

Flowcode est un logiciel graphique de programmation spécialisé pour les µcontrôleurs de la famille PIC du fabriquant *Microchip*.

Les graphiques utilisés par ce logiciel sont des algorigrammes ou ordinogrammes, ils permettent la simulation du programme et la compilation du programme en code Hexadécimal (.Hex) compréhensible par le PIC.
Note : Il est également possible d'obtenir un code en langage C compréhensible par l'homme.



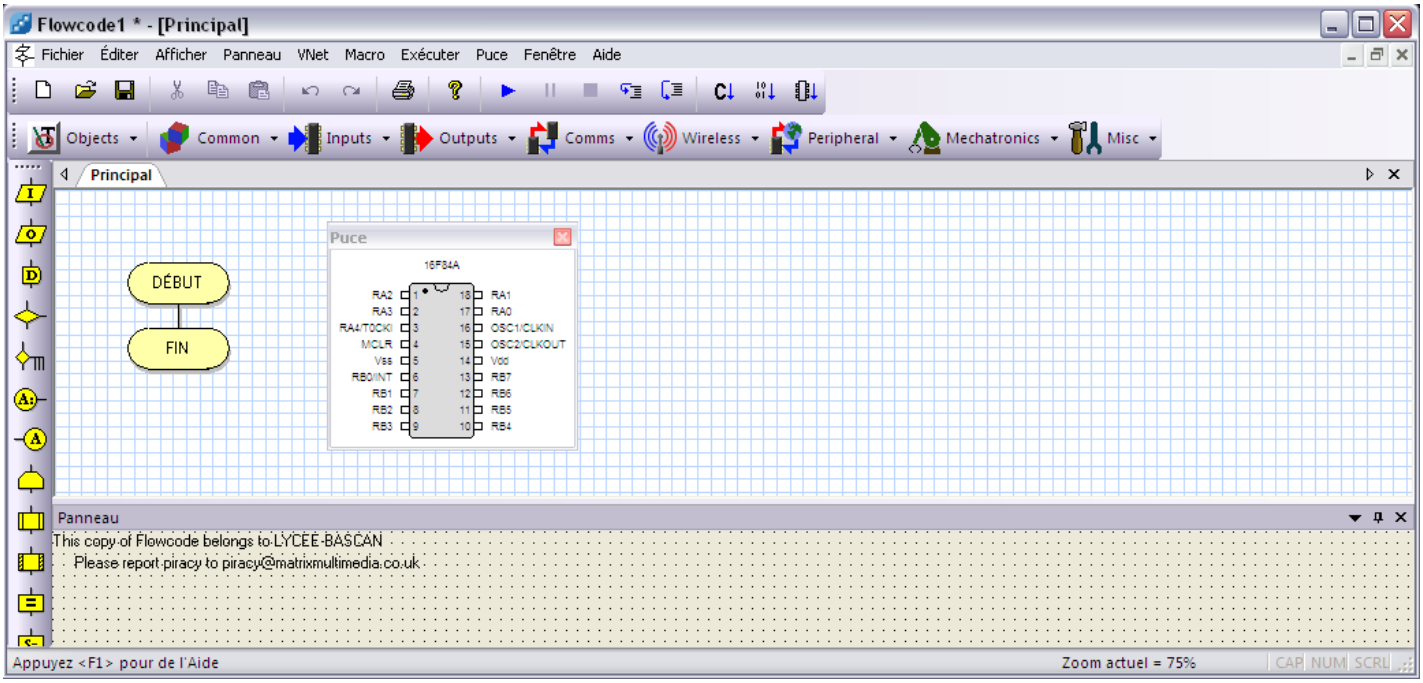
2. Travail demandé :

2.1. Réalisation du premier programme :

2.1.1. Affectation des entrées/sorties :

- Lancer le logiciel ***FLOWCODE V4***.
- Cliquer sur OK à l'invite "créer un nouvel algorigramme FlowCode".
- Choisir le µcontrôleur 16F84A

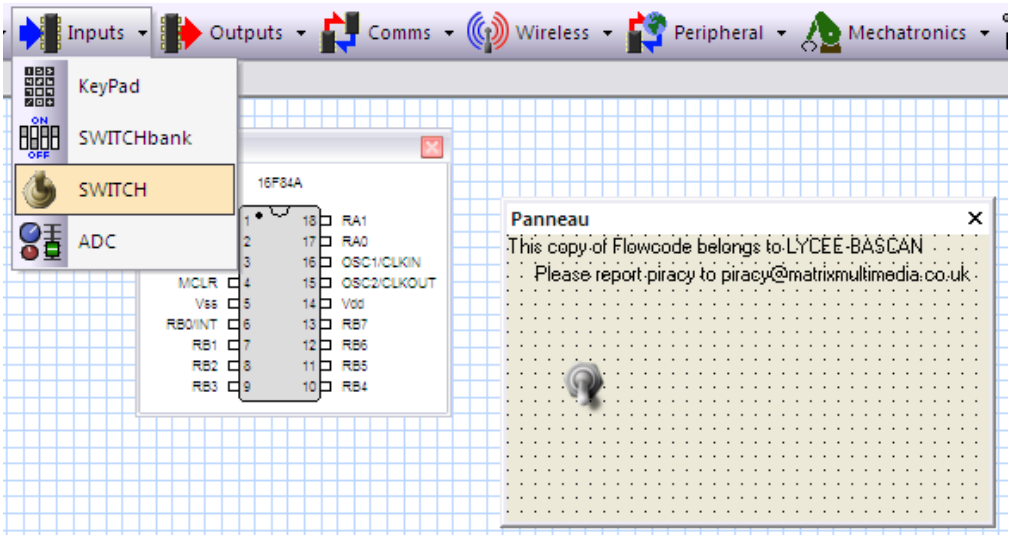
Vous obtenez la page suivante à l'écran :



Dans ce premier exercice nous allons réaliser un simple interrupteur, qui allume une LED.

L'interrupteur S0 sera raccordé sur le bit 0 du port B (RB0)
La LED1 sera raccordée sur le bit 1 du port A (RA1)

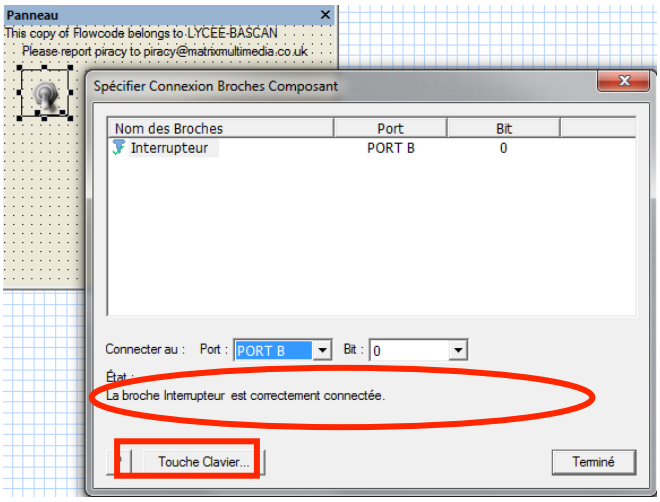
Insérer un interrupteur dans le **panneau** à l'aide du menu déroulant **"Inputs".**



Affecter l'interrupteur S0 au bit 0 du port B (RB0).
Clic droit sur le "switch", choisir ***"connexions".***

Note : Pour plus de praticité lors de la simulation, vous pouvez affecter cet interrupteur à une touche du clavier.

Affecter en suivant la même procédure une **DEL** rouge au **bit 1 du PORTA** (RA1).



2.1.2. Recherche de l'algorithme :

Pour écrire le programme, il faut traduire sous forme d'algorigramme la phrase suivante :

Si l'interrupteur S0 est actionné alors on allume la DEL1, sinon, on éteint la DEL1.

Il faut affiner la réflexion car le µcontrôleur ne comprend que les 0 et les 1 logiques et déclarer les variables que l'on utilise S0 et DEL1 :

Si S0=1 (actionné) alors DEL1 = 1 (allumée)
sinon DEL1 = 0 (éteinte)

Et tenir compte des affectations d'entrées/sortie afin que le µcontrôleur "comprenne" que la variable d'entrée S0 est affectée au bit 0 du port B et que la variable de sortie LED1 est affectée au bit 1 du port A.

```
S0 = RB0              // On charge dans le variable S0 la valeur du bit 0 du port B
Si S0 = 1 alors       // On effectue un test sur le contenu de la variable S0
      DEL1 = 1        // On écrit la valeur souhaitée dans la variable DEL1 en fonction du test
Sinon
      DEL1=0          // On écrit la valeur souhaitée dans la variable DEL1 en fonction du test
RA1 = DEL1            // Le bit 1 du port A prend la valeur de la variable DEL1.
```

Q1. Quelles sont les valeurs que peut prendre la variable S0 ?

Q2. Peut-on choisir d'autres valeurs que 0 ou 1 pour la variable DEL1 ?

- **Traduction de l'algorithme en algorigramme :**

Pour réaliser l'algorigramme, il faut utiliser :

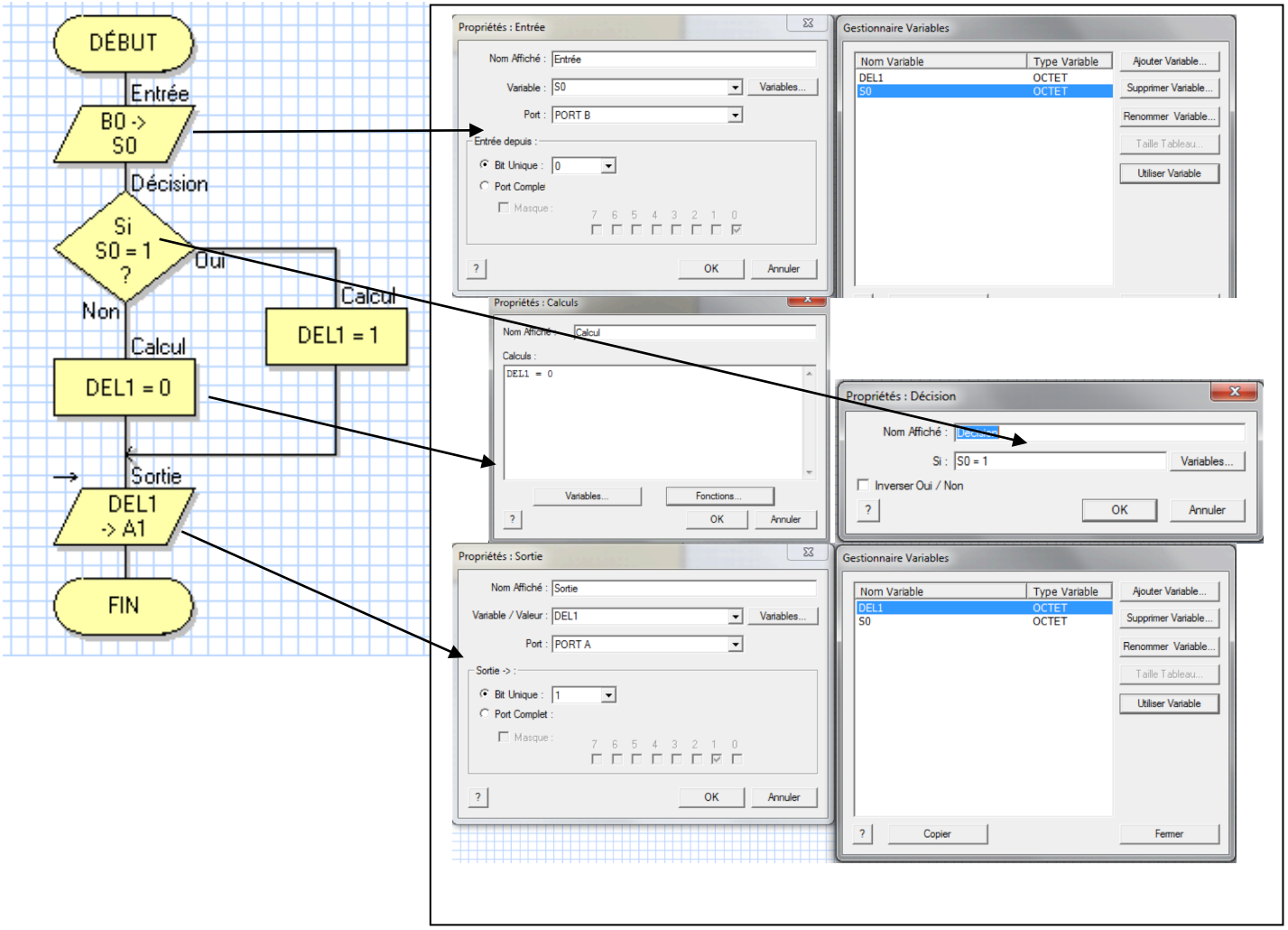
une icône d'entrée

une icône de sortie


une icône de décision

des icônes de calcul

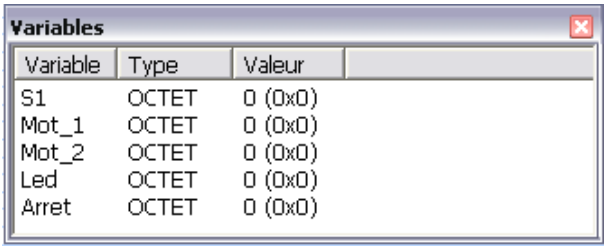
Etablir l'algorigramme suivant (accès aux différents pictogrammes par clic droit) et **insérer les commentaires nécessaires à sa compréhension :**



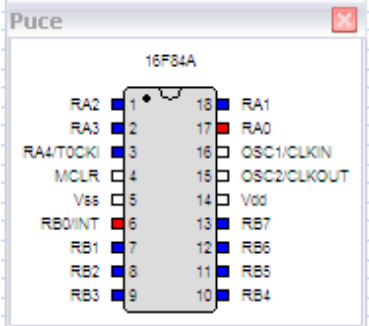
2.1.3. Simulation :

Exécuter la simulation en exécutant pas à pas les instructions du programme grâce au bouton :

Pour vous aidez à vérifier le bon fonctionnement de votre programme, ajouter les variables utilisées dans la fenêtre "Variables", ce qui vous permettra de vérifier en permanence leur état (**clic droit / ajouter variables …**) :

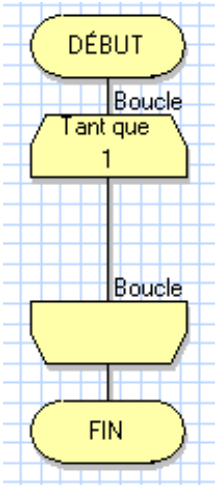


Vous pouvez également voir l'état des ports dans la fenêtre "Puce" :
Exemple : ici RA0 et RB0 sont à 1 (en **rouge**) et les autres ports sont à 0 (en **bleu**).

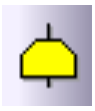


Q3. Que constatez-vous ? Le fonctionnement correspond t-il à celui attendu ?

Comme vous l'avez remarqué, le programme ne fonctionne pas !!!!!!



Et oui, le programme ne s'exécute qu'une fois, si l'interrupteur n'était pas actionné la DEL1 est restée éteinte, et s'il était actionné, on ne peut plus l'éteindre.

Il faut ajouter une boucle infinie à ce programme, avec l'icône ci-contre : 

Réaliser la modification, exécuter le programme en simulation :  .

2.2. Réalisation d'un clignoteur :

En reprenant la structure de l'algorigramme précédent, concevoir le programme tel que :

Si l'interrupteur S0 est actionné, la DEL1 doit clignoter avec un allumage de 400ms, une extinction de 200ms, sinon, la DEL1 reste éteinte.

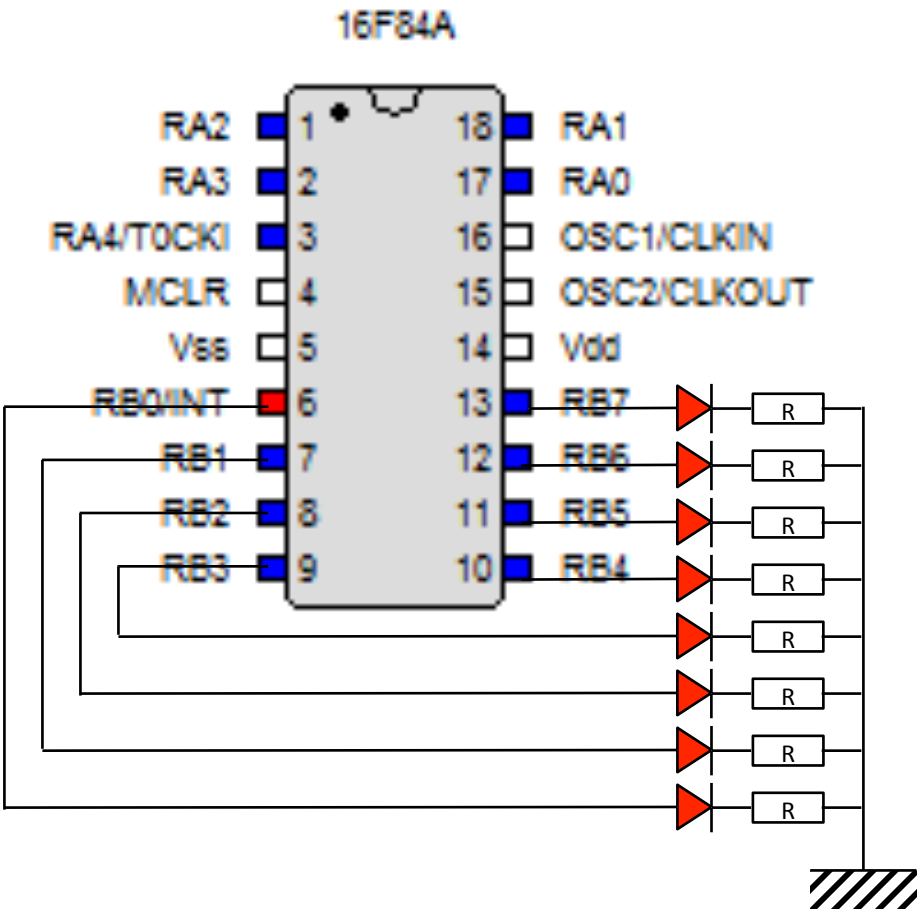
Note : Fonction temporisation à l'aide du pictogramme pause. 

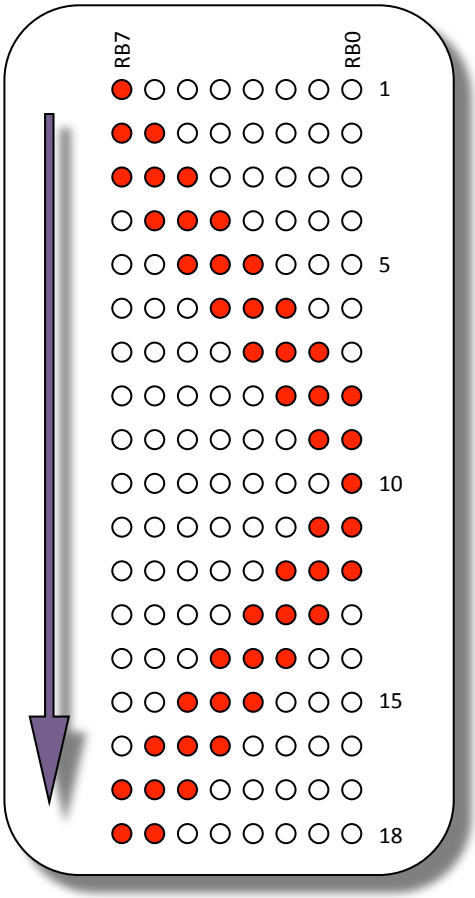
Q4. Modifier l'algorigramme précédent et insérer les commentaires nécessaires à sa compréhension.

Q5. Réaliser la simulation et la faire valider.

3. Chenillard :

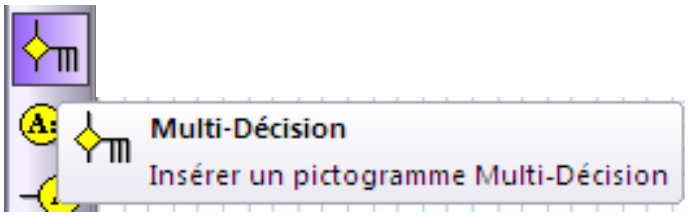
Nous allons maintenant réaliser la **programmation** d'un **chenillard** :
Les LEDS sont connectées comme suit sur le Port B et on donne le **cycle qui devra être répété indéfiniment** :





3.1. Conseils pour la conception :

- Utiliser une structure **multi-décision**.
- Travailler sur les **octets** et non plus uniquement sur les bits.
- Utiliser une **fonction** (macro) pour l'allumage des différentes LEDS.

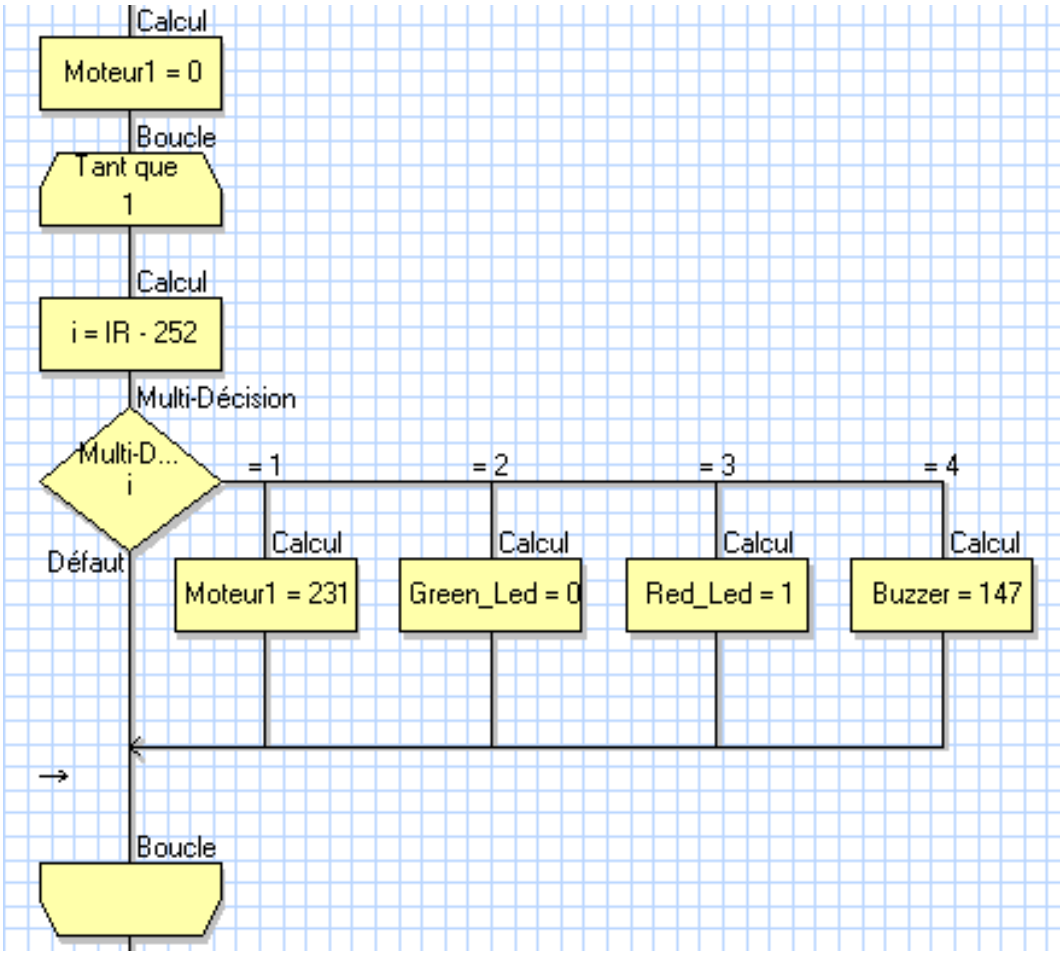


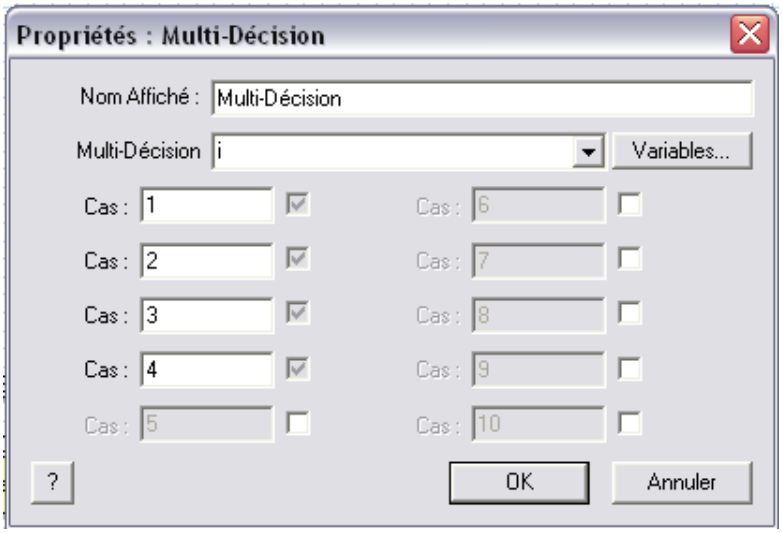
Multi-décision :

Utiliser une variable qui s'incrémente (+1) ou se décrémente (-1) à l'aide de la fonction calcul. En fonction de la valeur de cette variable, allumer certaines LEDS (mot binaire à appliquer sur le port) grâce à la fonction multi-décision.

Dans l'exemple ci-dessous, la variable s'appelle " **i** " et peut prendre les valeurs de 1 à 4.

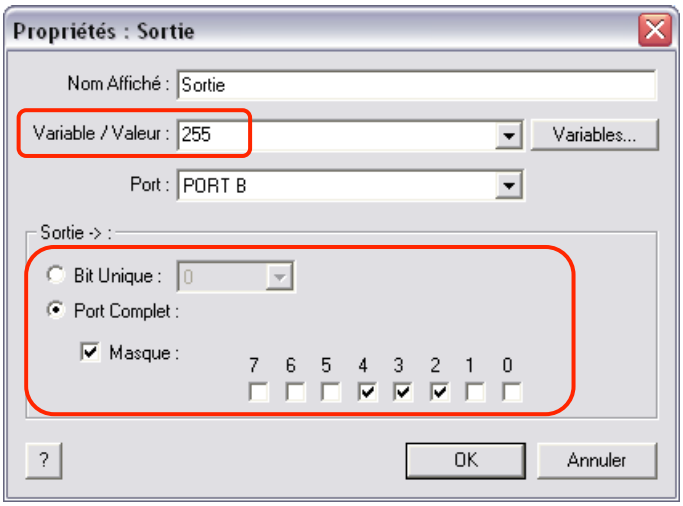
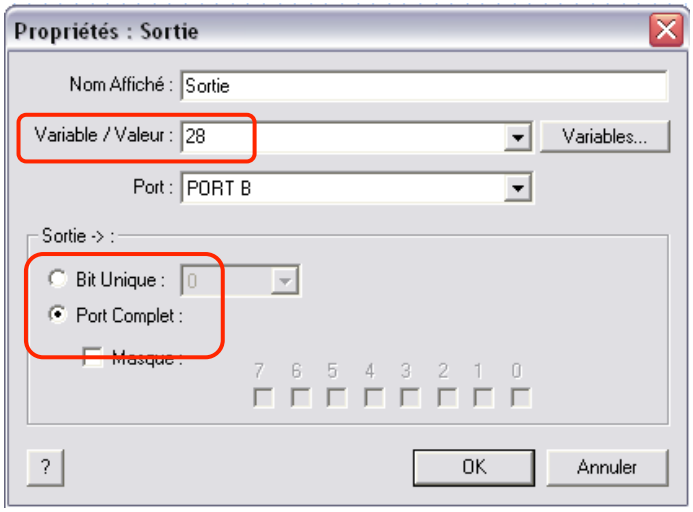
En fonction de sa valeur, une décision est prise.





3.2. Travail sur les octets :

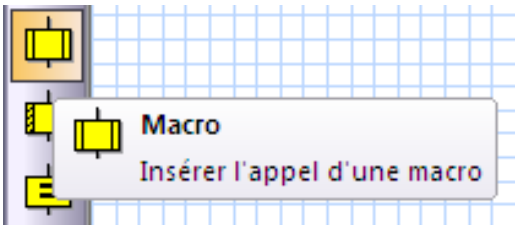
Exemple pour la ligne 6 ou la ligne 14 du cycle :

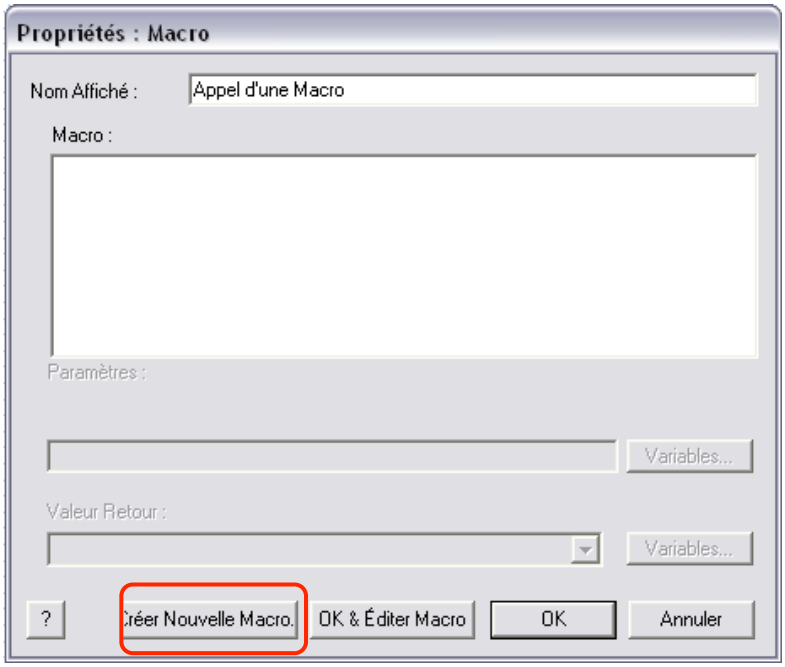
1ère méthode :	2ème méthode (à privilégier) :
On met tous les bits à 1 (valeur 255) et on utilise un masque.	On met à 1 uniquement les bits que l'on souhaite sans utiliser de masque, uniquement par calcul de la valeur de l'octet.
	

Rappel sur la numération								
Bit	a7	a6	a5	a4	a3	a2	a1	a0
Poids	7	6	5	4	3	2	1	0
Valeur décimal	$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
Exemple : Lignes 6 et 14.	0	0	0	1	1	1	0	0
	4 + 8 + 16 = 28 Donc il faudra imposer une valeur de 28 au port B complet.							

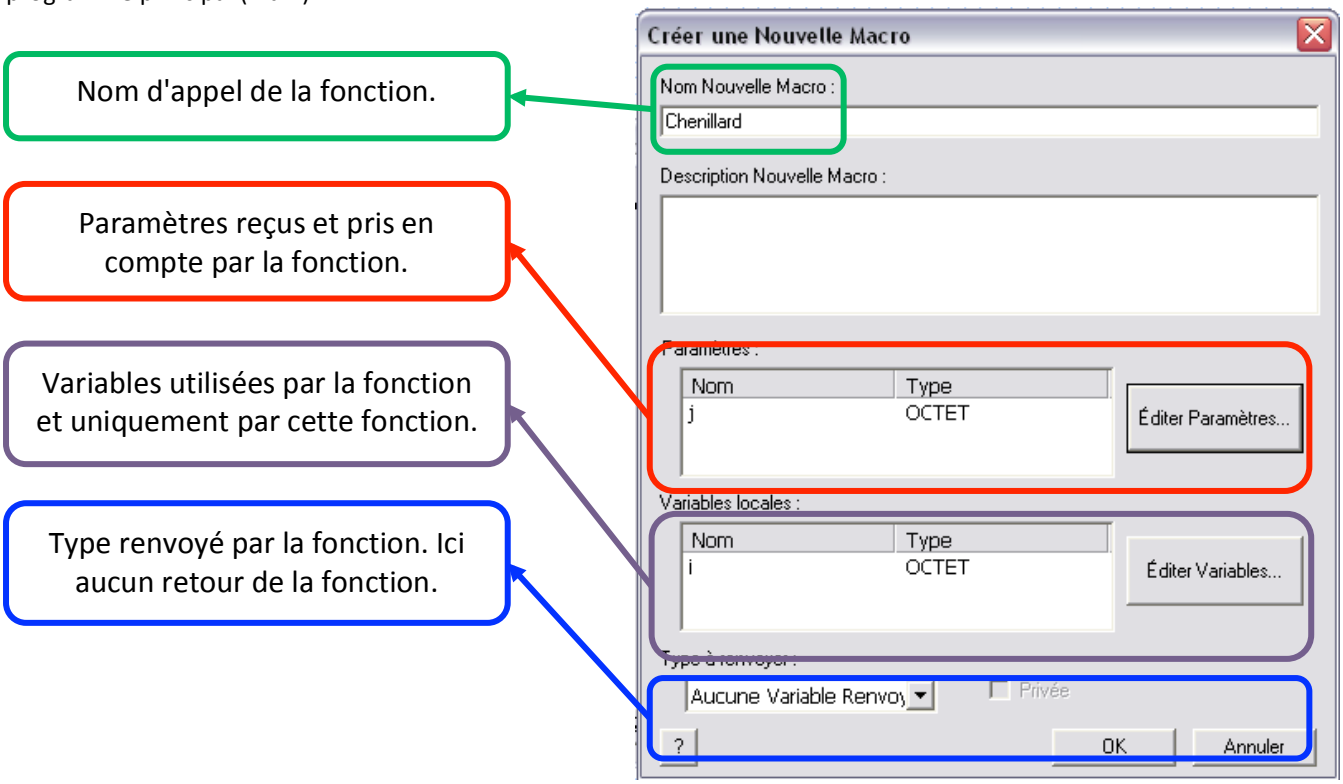
3.3. Création d'une macro (fonction) :

Afin d'améliorer la lisibilité du code nous allons utiliser une macro qui se chargera d'allumer les LEDS en fonction de l'évolution du programme principal.

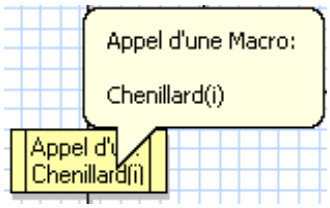


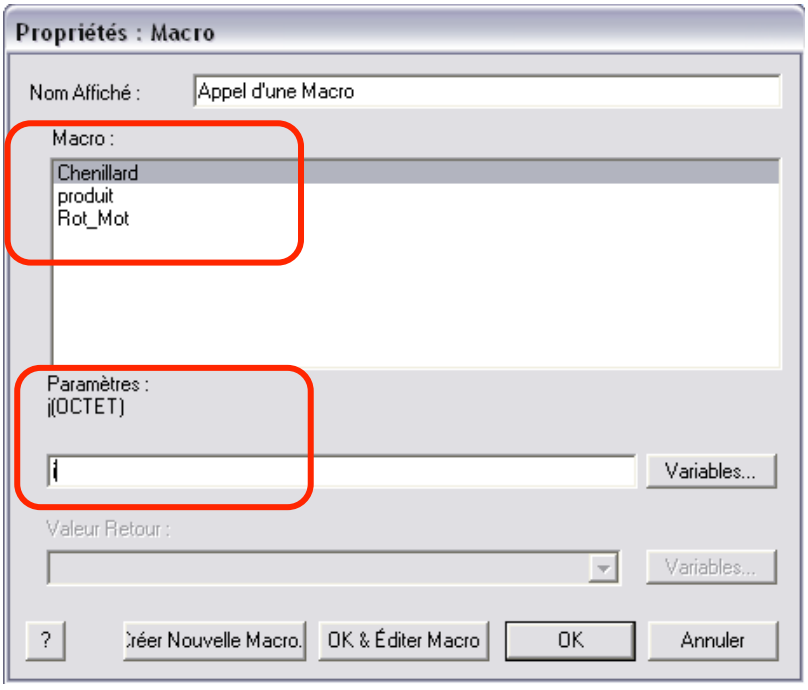


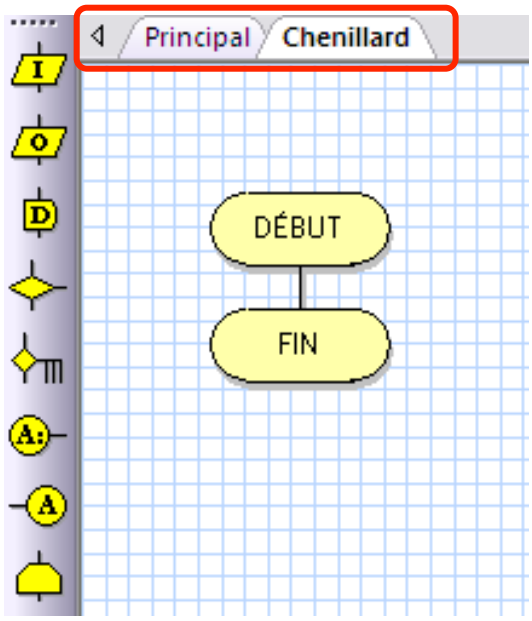
Une macro est une fonction (sous-programme), qui peut être appelée à n'importe quel moment lors de l'exécution du programme principal (main).



Une fois la macro créée, il suffit de la sélectionner dans la liste et de rentrer le paramètre qui sera pris en compte par la fonction (ici " **i** ")







Q6. ***Réaliser l'algorigramme, insérer les commentaires nécessaires à sa compréhension et faire la simulation.***

Q7. ***Imprimer votre algorigramme et la macro associée.***