

# COURS PIC16F628A

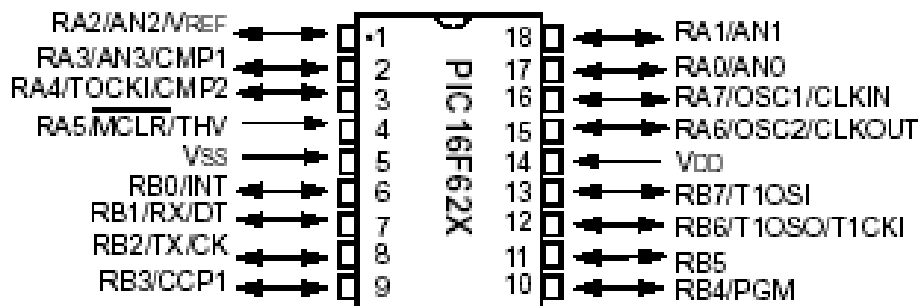
## PREMIERE UTILISATION DU MICROCONTROLEUR

### 1 Aspect matériel

Le microcontrôleur PIC 16F628A est un des modèles (le plus courant et un des plus petits) de la famille des circuits microcontrôleurs de Microchip. Présenté en boîtier DIL (Dual In Line) 18 broches, il possède de nombreuses et performantes caractéristiques. Les principales sont :

- une programmation facile en langage Basic ou en assembleur
- son faible coût,
- la possibilité d'être programmé insitu
- un oscillateur interne de 4MHz ou externe. Pour un besoin de précision ou de fréquence différente de 4MHz et jusque maximum 20MHz, il faut ajouter un quartz ou un circuit RC
- 16 lignes d'Entrées / Sorties :
  - ✓ 8 lignes sur le portA (avec possibilité sur 4 entrées de convertir sa valeur analogique en un mot numérique)
  - ✓ 8 lignes sur le portB
- architecture RISC (Reduced Instructions Set Computer : jeu d'instructions réduit qui convient à de petits programmes)
- une mémoire programme contenant 2048 instructions (codées sur 14 bits)
- une mémoire RAM de données de 224 octets
- une mémoire EEPROM de 128 octets
- des temporisateurs et un chien de garde
- une interface de transmission série

#### Dénomination des broches dans un boîtier DIL 18 broches



On remarquera que toutes les broches du circuit (sauf Vss, Vdd et RB5) ont plusieurs fonctions. Le choix de la fonction utilisée se fait par programmation.

On distingue en particulier :

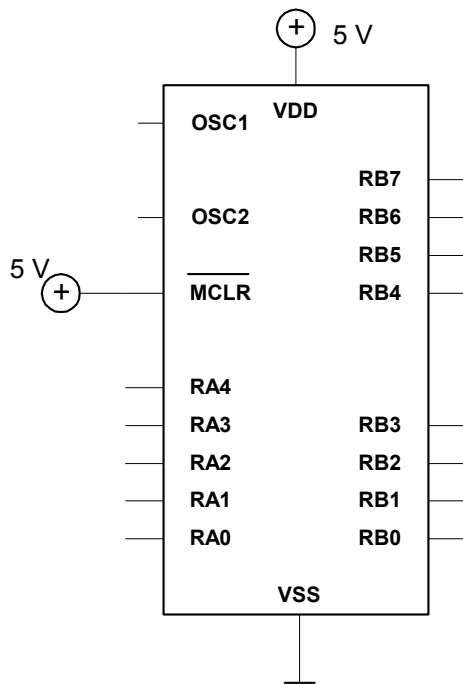
- VDD et VSS représentent respectivement l'alimentation 5V et 0V du circuit.
- Les deux broches OSC1 et OSC2 servent à recevoir si besoin le quartz destiné à l'oscillateur.
- MCLR (Master CLeAr) active à 0 correspond à l'entrée d'Initialisation (Reset) du circuit.
- RB0 à RB7 représentent les 8 lignes du port B et RA0 à RA7 les 8 lignes du port A, pouvant toutes être définies par programmation en entrées ou en sorties.

## 2 Développement d'une application

### 2.1 Principe

Le développement d'une application commence toujours par un cahier des charges qui définit l'électronique à implanter autour du microcontrôleur.

Vient ensuite l'élaboration de l'organigramme (ou de l'algorithme) puis la rédaction du programme en Basic dont le **fichier objet** (fichier hexadécimal traduisant le programme élaboré en langage Basic ) sera implanté dans le circuit.



Dans nos études simples le schéma de base sera toujours celui ci-contre.

Il faut placer les composants réalisant les Entrées ou les Sorties, comme par exemple des interrupteurs en entrées et des leds en sorties.

Quant à l'écriture d'un programme, elle passe invariablement par les étapes suivantes :

- a) Rédiger un programme en langage évolué à l'aide d'un éditeur de texte.

Dans notre exemple : le langage évolué sera le Basic et l'éditeur de texte sera celui du logiciel MikroBasic, il sera créé un fichier à extension **.pbas**

- b) Compiler le programme basic en un programme source à l'aide du compilateur basic qui créera un fichier **.asm**,
- c) Assembler le programme qui créera le fichier objet **.hex**
- d) Transférer ce fichier hexadécimal dans le microcontrôleur à l'aide d'un programmeur (carte électronique reliée au PC sur laquelle est placé le PIC à programmer) et d'un logiciel. Le logiciel ICPROG par exemple.
- e) Enfin placer le PIC programmé sur le montage qui lui est dédié.

*Remarque : Les étapes b) et c) se font sur un seul clic.*

# COURS PIC16F628A

## EXEMPLE D'ECRITURES D'UN PROGRAMME DANS SES DIFFERENTES EXTENSIONS

### a) Programme extension .pbas

```
program PIC_Store_Complet
    ' ***** déclarations *****
symbol descente = PortA.0
symbol montee   = PortA.1
symbol soleil   = PortA.2
symbol vent     = PortA.3
symbol mode     = PortA.4
dim Sortie,Dav,Mav,D,M      as byte

main:
    '*****      Main      *****
    CMCON      = 07          ' pour inhiber les comparateurs analogiques du 16F628
    OPTION_REG = $80        ' 1000 0001 = Pas de pull-up
    trisb      = 0          ' 0000 0000 = PortB en sortie
    trisa      = $FF        ' 1111 1111 = PortA en entrée
    PortB      = 0          ' aucune sortie active
    Dav = 0
    Mav = 0
    while true
        if mode = 0 then
            ' en mode manuel
            D = (Dav OR descente) And (not(montee))
            ' équation de M et D
            M = (Mav OR montee) And (not(descente))
            Dav = D
            Mav = M
        else
            ' en mode auto
            D = (soleil) And (not(vent))
            ' équation de M et D
            M = not(D)
            Dav = 0
            Mav = 0
        end if
        Sortie = D + 2 * M
        ' Mise à jour du portB
        PortB = Sortie
    wend
end.
```

### b) Programme extension .asm

```
;// ASM code generated by mikroVirtualMachine for PIC - V. 3.0.0.0
;// Date/Time: 20/09/2005 10:09:28
;// Info: http://www.mikroelektronika.co.yu
```

```
;// ADDRESS   OPCODE   ASM
;-----
$0000 $281A GOTO PIC_Store_Complet_main
$0004 $      math_mul_8x8_u:
$0004 $1303 BCF STATUS,RP1
$0005 $1283 BCF STATUS,RP0
$0006 $3008 MOVLW 8
$0007 $00A4 MOVWF MATH_MAIN_GLOBAL_LOOPCOUNT
$0008 $0822 MOVF MATH_MAIN_GLOBAL_X_1,W
$0009 $0CA0 RRF MATH_MAIN_GLOBAL_Y_1,F
$000A $1803 BTFSK STATUS,C
```

# COURS PIC16F628A

```
$000B $2810 GOTO $+5
$000C $0BA4 DECFSZ MATH_MAIN_GLOBAL_LOOPCOUNT,F
$000D $2809 GOTO $-4
$000E $01A2 CLRF MATH_MAIN_GLOBAL_X_1,F
$000F $3400 RETLW 0
$0010 $1003 BCF STATUS,C
$0011 $2815 GOTO $+4
ect.....
```

ect.....

```
$0073 $0823 MOVF Math_main_global_X_2, W
$0074 $00A9 MOVWF main_global_sortie
$0075 $0827 MOVF main_global_d, W
$0076 $07A9 ADDWF main_global_sortie, F
$0077 $0829 MOVF main_global_sortie, W
$0078 $0086 MOVWF PORTB
$0079 $2828 GOTO PIC_Store_Complet_L_0
$007A $ PIC_Store_Complet_L_2:
$007A $ PIC_Store_Complet_L_9:
$007A $287A GOTO PIC_Store_Complet_L_9
```

## c) Programme extension .hex

```
:100000001A28FF3FFF3FFF3F031383120830A4006D
:100010002208A00C03181028A40B0928A201003400
:1000200003101528A00C0318A207A20CA30CA40B04
:10003000122808000730031383129F0080308316B4
:1000400081008601FF30850083128601A501A6018B
:100050000030051A0130F20000307202031D33280F
:10006000FF30F3003428F301FF307302031D5928D9
:10007000003005180130F10025087104A700003098
:1000800085180130F300F3097308A70500308518BF
:100090000130F10026087104A80000300518013075
:1000A000F300F3097308A8052708A5002808A6008F
:1000B0006C28003005190130F1000030851901303D
:1000C000F300F30971087305A7002708F100F1098F
:1000D0007108A800A501A6012808A000A10102300E
:1000E000A200A30104202308A9002708A9072908C2
:1000F000860028287A28FF3FFF3FFF3FFF3FFF3F52
:02400E00013F70
:00000001FF
```

On y retrouve le code machine de chaque instruction en assembleur (partie grisée) mais écrite octet par octet à l'envers

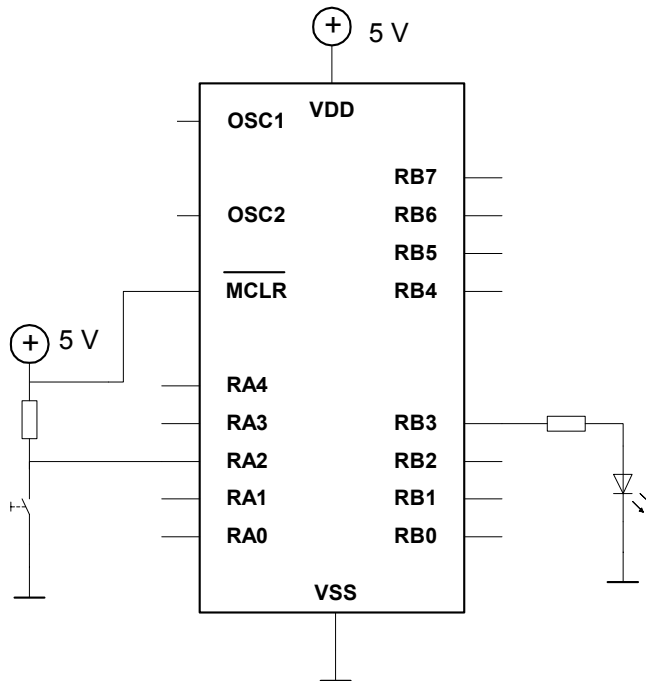
### Exemple :

Dans le fichier <b>.asm</b> la ligne	<i>\$0000 \$ 281A</i>	<i>GOTO PIC_store_complet_main</i>
correspond dans le fichier <b>.hex</b> à	<i>1A28</i>	
Ou encore la ligne	<i>\$0006 \$3008</i>	<i>MOVLW 8</i>
correspond dans le fichier <b>.hex</b> à	<i>0830</i>	
Dans le fichier <b>.asm</b> la ligne	<i>\$007A \$ 287A</i>	<i>GOTO PIC_store_complet_L_9</i>
correspond dans le fichier <b>.hex</b> à	<i>7A28</i>	

# COURS PIC16F628A

## 2.2 Un premier exemple

Dans le montage suivant, l'appui sur un bouton poussoir provoque l'allumage définitif d'une led, initialement éteinte.



### program Exemple\_1\_cours

```

symbol bp = PortA.2      ' Déclarations de variables
symbol led = PortB.3

main:                    '***** Début du programme principal *****
    cmcon=07              'inhibition des entrées analogiques du port A
    trisa = $FF           ' 1111 1111 = PortA.2 en entrée
    trisb = 0             ' 0000 0000 = PortB en sortie
    portb = 0            ' mettre 0 sur toutes les sorties

    while bp = 1         ' des que bp = 0 (appui),
    wend                 ' on sort de la boucle
    led = 1              ' la sortie RB3 passe à 1, la led s'allume

end.                    'indique au logiciel la fin du programme à compiler et à assembler
    
```

### Explications supplémentaires :

Pour une utilisation et une relecture plus simples du programme, on utilise couramment des alias dont il faut en tout premier lieu déclarer l'existence et le type: ici led et bp

On commence toujours le programme principal par la configuration des ports A et B (définition pour chaque broche de chaque port de l'utilisation en entrée ou en sortie). Ce qui nécessite d'écrire dans les registres TRIS A et TRIS B :(1) pour entrée ou (0) pour sortie

On utilise ici les entrées du portA comme étant des valeurs numériques, il faut par conséquent écrire 07 dans le registre CMCON



## 3 Développement avec le logiciel MikroBasic (ver 2.0)

### 3.1 Les principales instructions

**Les opérateurs arithmétiques élémentaires :**

- + : addition
- - : soustraction
- \* multiplication
- / : division
- div : exécute la division et restitue la partie entière
- mod : exécute la division et restitue le reste de la division

**Les opérateurs logiques élémentaires :**

- & : (AND)
- ! : (OR)
- ^ : (XOR)
- not : inverseur
- >> x : où x indique le nombre de décalages à droite successifs dans un mot binaire
- << x : où x indique le nombre de décalages à gauche successifs dans un mot binaire.

**Les principales instructions :**

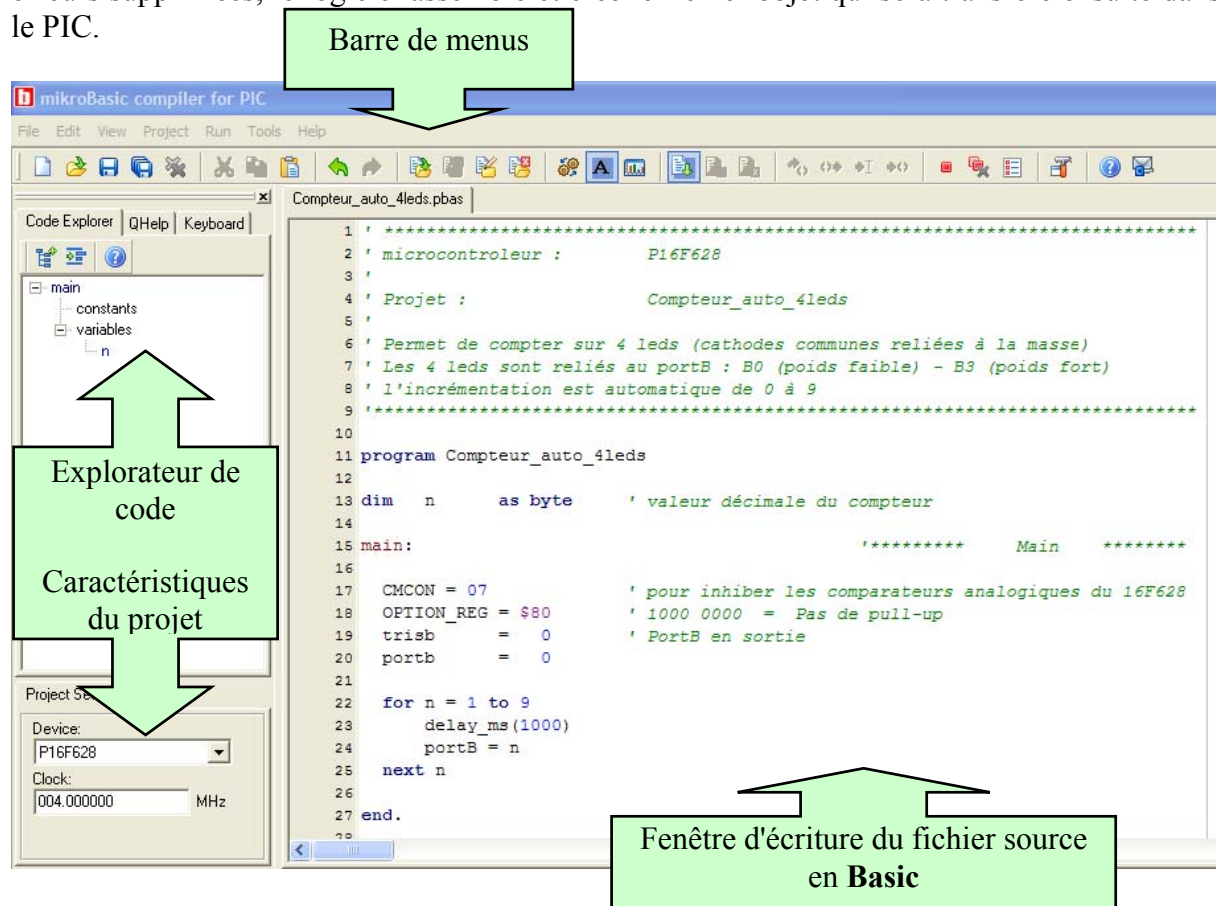
Syntaxe	Commentaires	Exemple
<b>program</b> ... <b>end.</b>	<b>Program</b> précise à la première ligne le nom du fichier en basic. La fin du programme est repérée par <b>end</b> avec un point	<b>program essai</b>  <b>end.</b>
<b>dim</b>  <b>as byte</b> <b>(integer,...)</b>	Permet de déclarer les variables utilisées dans le programme en précisant leur type : <ul style="list-style-type: none"> <li>• byte : octet (8 bits)</li> <li>• integer : 16 bits</li> <li>• word : nom alphanumérique</li> </ul>	<b>dim i, j, k as byte</b> <b>dim counter as word</b> <b>dim tab as longint[100]</b>
<b>const</b>	Déclare une donnée constante de type numérique ou caractère	<b>const MIN = 1000</b> <b>const SWITCH = "n"</b> <b>const vals as byte[12] = (31,12,17)</b>
<b>symbol</b>	Déclaration d'alias	<b>symbol t1s = delay_ms(1000)</b> <b>symbol led = PortB.3</b>
<b>sub</b> ... <b>end sub</b>	Déclaration des sous programmes (procédures ou fonctions) pour une meilleure structure du programme. S'écrivent avant le programme principal	<b>sub procedure calcul</b> <b>n = a * (b +3)</b> <b>end sub</b>
<b>main :</b>	Etiquette de début de programme principal, toujours suivi de deux points	<b>main :</b>
<b>if ... then ...</b> <b>(else) ...</b> <b>end if</b>	Structure de contrôle pour réaliser un test à l'aide d'une expression booléenne. Exécute un traitement si condition vraie (ou éventuellement un autre si faux)	<b>if plus = 1 then</b> <b>i = i+1</b> <b>else i=i*2</b> <b>end if</b>

# COURS PIC16F628A

<b>while ...</b> ... <b>wend</b>	Pour répéter un traitement tant qu'une condition est vraie.  Rem : s'utilise aussi pour créer une boucle sans fin	<b>while i &lt; 4</b> <b>i = i+1</b> <b>wend</b>  <b>while true</b> ... <b>wend</b>
<b>Select case</b> <b>case 0</b> ..... <b>case 3</b> ..... <b>case else</b> ..... <b>end select</b>	Suivant que la variable vaut 0 ou 3 ou autre → faire	<b>Select case j</b> <b>case 0</b> <b>portB=%00001111</b> <b>case 3</b> <b>portB=%01111111</b> <b>case else</b> <b>portB=0</b> <b>end select</b>
<b>for ... to ...</b>  <b>next</b>	Permet de réaliser une itération à l'aide d'une variable	<b>for i = 0 to 4</b> <b>portB = i</b> <b>next i</b>
<b>delay_us (n)</b> <b>delay_ms (m)</b>	Fonctions prêtes à l'emploi pour réaliser une temporisation de n microsecondes ou m millisecondes	<b>delay_ms (500)</b>
<b>goto</b>	Renvoi incondtionnel à une ligne de programme définie par une étiquette. L'étiquette est indiquée par son nom suivi de deux points ( :) <b>A éviter autant que possible.</b>	<b>goto main</b>

## 3.2 Utilisation du logiciel MikroBasic

Le logiciel MikroBasic est un environnement de développement intégré (IDE). Il est constitué entre autres d'un éditeur et d'un compilateur et de toutes commandes nécessaires à la compilation (création du fichier assembleur) et à l'assemblage (création du fichier objet en hexadécimal). Toute erreur de syntaxe est signalée par le compilateur et stoppe la compilation. Une aide contextuelle est disponible (voir paragraphe ci après). Une fois les erreurs supprimées, le logiciel assemble et crée le fichier objet qui sera transféré ensuite dans le PIC.



### Voici les étapes de développement de l'application :

- Création d'un projet (fichier.pbp). Il faut absolument un projet pour un programme.
- Rédaction du programme source en Basic dans la fenêtre principale.
- Enregistrement du programme source (fichier.pbas)
- Compilation (puis assemblage)
- Correction des éventuelles erreurs signalées par le compilateur et recompilation

Après exécution de ces étapes, nous disposons du fichier *.hex* à implanter dans la mémoire du microcontrôleur PIC.

Avant de programmer le circuit, il est conseillé de s'assurer du bon fonctionnement du programme par une simulation pas à pas associée à une visualisation des états des différentes variables et registres internes

### Remarque :

Ceci est une présentation succincte de MikroBasic. Lors de la création du programme, il est conseillé de lire la notice « Utilisation de MikroBasic ».

## 4 La programmation du composant

Elle nécessite l'emploi d'une carte de programmation à relier à un PC via une liaison série RS232 (COM1 ou 2 du PC). Le PC est équipé du logiciel de programmation ICPROG. La carte dispose d'un emplacement destiné à recevoir le composant à programmer.

### 4.1 La carte de programmation :

Voici la carte que nous utiliserons :



### 4.2 Programmation d'un PIC :

Il est impératif de faire dans l'ordre les manipulations suivantes :

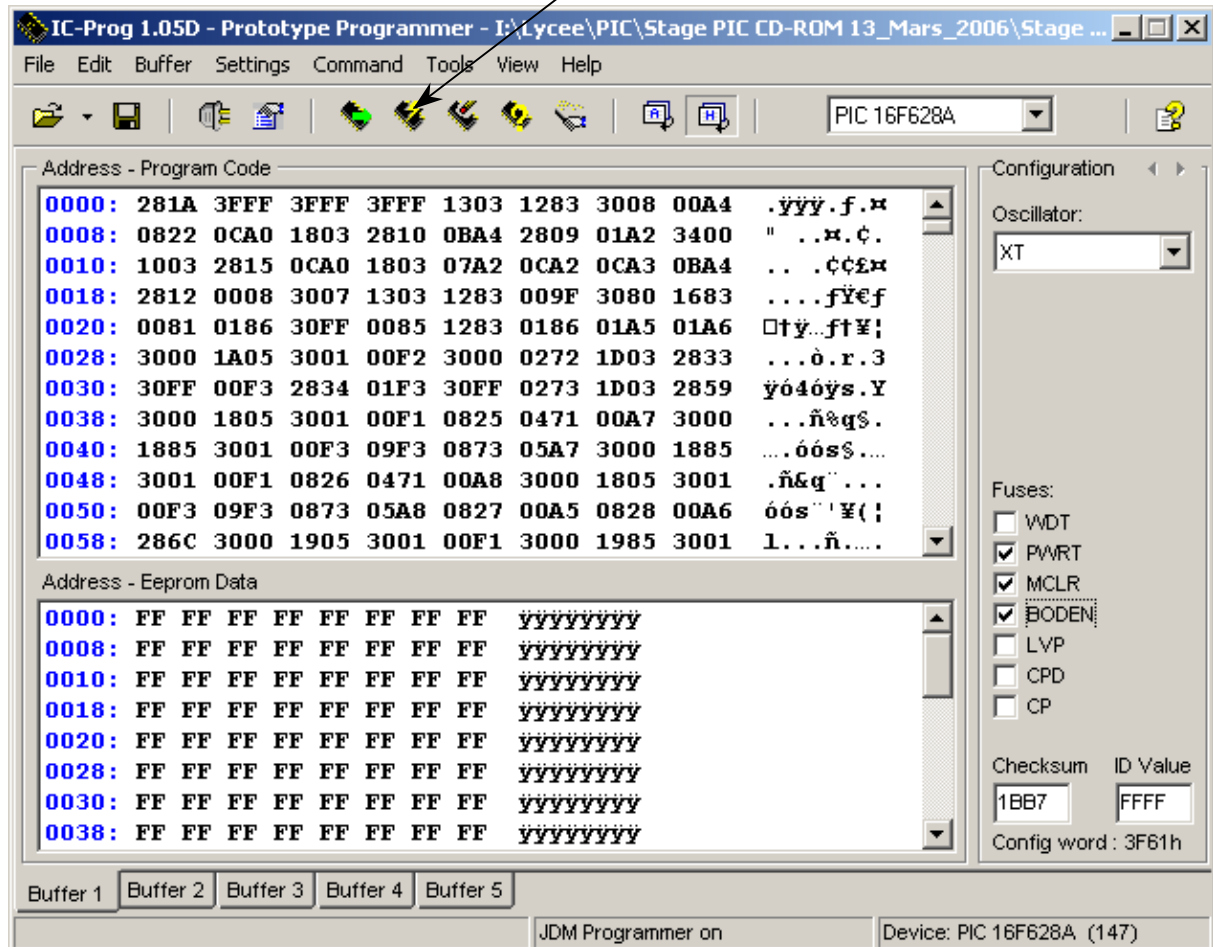
- Placer le circuit intégré PIC sur la carte de programmation
- Brancher la carte sur la liaison série d'un PC via le câble à connecteurs DB9
- Programmer avec le logiciel ICPROG :



- Lancer icprog.exe
- Ouvrir le fichier hexadécimal correspondant au programme à implanter dans le composant. Celui-ci apparaît dans la fenêtre. Il est possible de voir le programme en assembleur en cliquant sur A, ou en hexadécimal en cliquant sur H.
- Choisir le modèle du composant à programmer, 16F628A dans notre cas.
- Positionner les paramètres concernant la programmation (oscillateur, fusibles)

# COURS PIC16F628A

- Lancer la programmation par le menu **Commande** → **Tout Programmer** ou par la touche de fonction F5 ou par l'icône.



- d) Après la programmation, il faut débrancher la carte programmeur du PC
- e) Enlever le circuit intégré PIC.
- f) Placer le PIC sur son montage et tester le fonctionnement attendu

## Remarque :

Ceci est une présentation succincte de la programmation. Avant toute manipulation pour écrire dans le composant, il est conseillé de lire la notice « **Utilisation du programmeur** ».