AMICUS 18 (2ème partie)

Dans la première partie, nous avions présenté la platine Amicus 18 et nous avions réalisé quelques montages simples. Nous allons découvrir un peu mieux la programmation. Dans la séance précédente, nous n'avions entrevu que quelques fonctionnalités de sortie (allumer une diode, générer un son). Mais Amicus a plus d'un tour dans son sac et nous permet d'envisager de nombreuses fonctions d'entrée. Cet article est la synthèse de ce qui a été présenté par Vladimir F4FNA lors de la réunion du 08/01/11.

4) Présentation du logiciel Amicus IDE

Amicus IDE est un logiciel avec lequel on développe des programmes pour Amicus18. C'est un environnement intégré qui contient un éditeur de textes, le compilateur, des utilitaires et le programmateur pour Amicus18. L'installation du logiciel a déjà été présentée dans la première partie (paragraphe 2). Voyons plus précisément les fonctions disponibles.

Dans la partie haute de l'écran, on a la barre des menus et la plupart des icônes pour les diverses opérations d'édition et de compilation. Il y a aussi quelques « plug-ins » bien utiles : table de caractères ASCII, terminal de commande RS232 (Serial Com), calculatrice, etc. (voir image n°13)

Image n°13

Image n°14

Dans la partie gauche de l'écran on trouve le module « Code Explorer » (voir image n°14), qui nous donne la possibilité de voir et manipuler la structure de notre programme. Tous les éléments constitutifs du programme sont classés et on peut y accéder simplement en cliquant. Si on clique sur «t_1», le curseur de la fenêtre de l'éditeur de programme (image n° 15) pointe sur la ligne 3

Le nom du programme (ici «Allume_LED») est affiché dans l'en-tête. Chaque ligne est numérotée. D'autre part, les éléments du programme (variable, sorties, commentaires, etc.) sont de couleurs différentes pour une meilleure lisibilité.

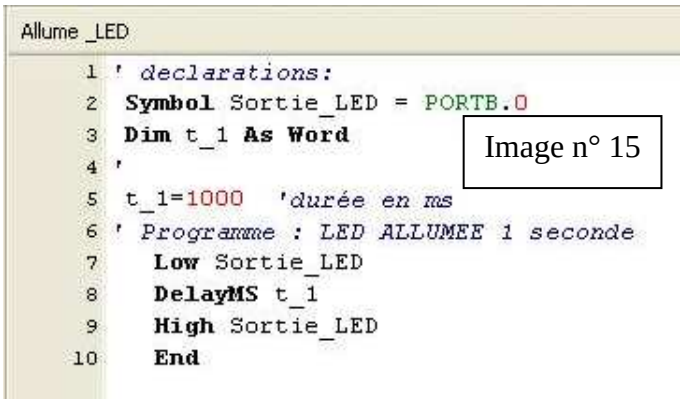


Image n° 15

Au bas de l'écran, le résultat de la compilation, une fois qu'elle a été lancée une première fois (voir image n°16), nous informe sur la quantité de mémoire utilisée, le nombre de variables utilisées et la version du programme.

Image n° 16

Un petit rappel de la programmation en langage Basic permet de mieux comprendre les termes employés dans la suite de cet exposé.

Système binaire, valeur définie en décimal ou héxadécimal

L'unité de base est le BIT (Binary digIT, chiffre binaire en français) qui a deux états possibles que l'on peut nommer différemment selon le contexte :

- 1 ou 0 ;
- VRAI ou FAUX ;
- HIGH ou LOW (état Haut ou état Bas), etc…

D'un point de vue électrique, 1 est toujours associé à une tension positive appelée Vdd (+ 3,3 V dans notre cas) et 0 à une tension nulle appelée GND (GrouND, la masse en français, soit 0 V)

Le PIC du Amicus18 fait partie de la très grande famille des microcontrôleurs 8 bits, donc les données sont traitées par tranches du même nombre, appelées BYTES (ou octets en français)

Un BYTE contient 8 bits organisés en deux NIBBLES (ou demi-octet) de 4 BITS : le premier NIBBLE est qualifié de « supérieur » et le second NIBBLE sera l'« inférieur ».

Un NIBBLE étant constitué de 4 bits, il peut avoir 16 valeurs possibles : 0 à 15 en décimal. Le système décimal ne compte que 10 chiffres pour former des nombres. Il va donc nous manquer 6 chiffres pour écrire un nombre en Hexadécimal (nombre en base 16) et nous emploierons pour cela les lettres A à F.

Un BYTE aura 256 valeurs possibles (16^2) : 0 à 255 en décimal ou 00 à FF en Héxadécimal.

Un WORD (mot en anglais) contient 16 bits organisés en deux BYTES. Là encore, on distinguera le BYTE « supérieur » (le premier) et le BYTE « inférieur » (le second). Un WORD peut avoir 65536 (256^2) valeurs possibles : 0 à 65535 en décimal ou 0000 à FFFF en Hexadécimal

Un DWORD (dubble word) continent 32 bits organisés en quatre BYTES ou deux WORDS. La plage des valeurs est : 0 à 4294967295 (65536^2 -1) en décimal ou 00000000 à FFFFFFFF en Héxadécimal.

Identifiant

Un identifiant est le terme technique utilisé pour nommer un élément d'un programme. Les identifiants sont utilisés pour nommer les étiquettes, les variables, les constantes et les alias.

Un identifiant est une séquence de lettres, chiffres et underscores (trait de soulignement ou « tiret du 8») ; l'identifiant ne commence jamais par un chiffre. Les identifiants ne tiennent pas compte de la casse des lettres (majuscules/minuscules). Par conséquent "label", "LABEL", et "Label" sont tous traités comme le même mot. Les caractères accentués ne sont pas non plus différenciés. De plus, l'espace est considéré comme un caractère mais est interdit. On le remplacera dans les identifiants par l'underscore, comme dans « t_1 » vu précédemment. Les étiquettes peuvent avoir n'importe quel nombre de caractères, seuls les 32 premiers sont reconnus. Ainsi « Identifiant_pour_definir_la_variable_1 » et « IDENTIFIANT_pour_definir_la_variable_2 » ne pourront pas être différenciés par le logiciel.

Etiquette (Label)

Les étiquettes (label en anglais) sont des identifiants suivis de deux points sans espace (exemple "etiquette:"). Ces etiquettes servent à signifier au compilateur les destinations des commandes **GoTo**, **Call** ou **GoSub.**

Exemple: LED clignotant par période 1 seconde en boucle sans fin

```
    Etiquette_01:           'début
            Low PORTB.0     'allumer la LED en mettant le port B à la masse
            DelayMS 500     'attendre 500ms
            High PORTB.0    'éteindre la LED en mettant le port B à +3,3 V
            DelayMS 500     'attendre 500ms
```

```
GoTo Etiquette_01 'va au début (au label “etiquette”)
```

On peut créer des étiquettes à tout moment. Les étiquettes sont répertoriées dans le pavé « Code Explorer » du logiciel Amicus IDE sous l’onglet « Labels ».

Variable

La variable est un élément du programme dont la valeur peut changer en fonction du déroulement de celui-ci. Ceci la différencie de la constante qui, elle, est définie une fois pour toute. On peut créer une variable en utilisant le mot clé DIM au début du programme. Il est important de bien choisir le type de variable en fonction de la plage des valeurs nécessaire.

Les variables peuvent être de 5 types : Bit, Byte, Word, Dword ou FLOAT (à virgule flottante). L'espace de mémoire pour chaque variable est automatiquement affecté dans la zone RAM (mémoire vive) du microcontrôleur. Les variables peuvent être de type "non-signées" (la valeur 0 est le minimum de la plage des valeurs) ou "signées" (la valeur 0 est située au milieu de la plage des valeurs).

Le format pour la création d'une variable est le suivant :

Dim Label **as** Size

“Label” est l’identifiant de la variable (“Label” fait partie des Mots-clés exclus, voir plus loin). “Size” peut prendre les valeurs suivantes : Bit, Byte, Word, Dword ou Float.

Pour illustre notre propos, voici quelques exemples de créations des variables:

```
Dim Octet as Byte              ' Création d'une variable octet   (0 à 255)
Dim Etat as Bit                ' Création d'une variable bit (0 ou 1)
Dim Double_octet as Word       ' Création d'une variable 16 bits  (0 to 65535)
Dim Grand_Entier as Dword      ' Création d'une variable 32 bits signés (-2147483648 à
                                 +2147483647)
Dim Nombre_a_virgule as Float  ' Création d'une variable 32 bits à virgule flottante
```

Concernant les variables Dword, on peut les déclarer aussi non-signées. Dans ce cas, la plage des valeurs commence à 0 (et non à -2147483648) et finit à 4294967295. Pour ce faire, il faut mettre une declaration de type :

```
Unsigned_Dwords = On
```

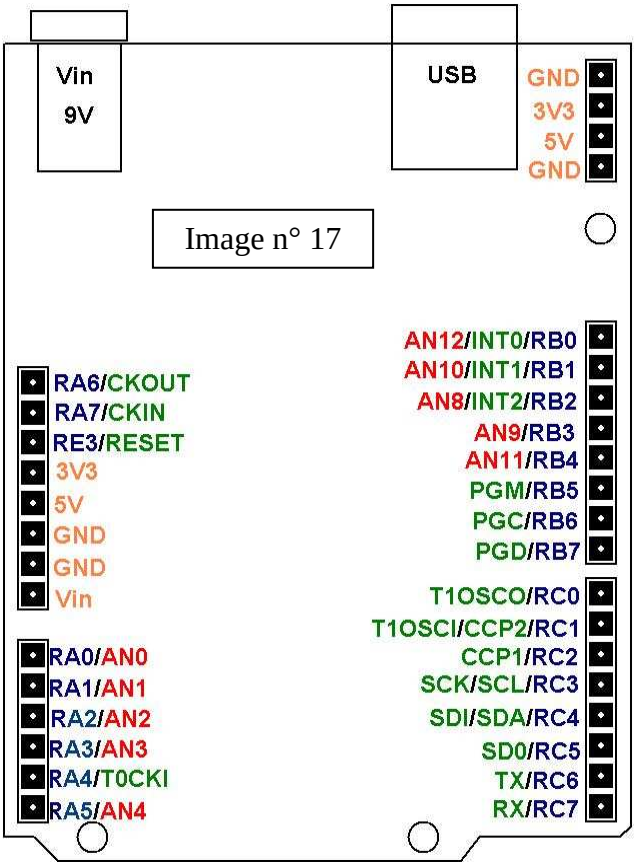
Le nombre des variables possibles dépend de la quantité de mémoire RAM du microcontrôleur et aussi des types de variables du programme BASIC. Notez que le compilateur va créer des variables Système additionnelles nécessaires aux calculs des équations complexes ou des structures des commandes complexes, surtout pour les calculs en virgule flottante.

Aussi, il y a certains identifiants réservés qui ne peuvent pas être utilisés comme noms de variables. Ce sont les variables du système utilisées par le compilateur. Les mots suivants ne doivent pas être utilisés comme noms de variables, sinon le compilateur ne peut pas créer ces noms en cas de besoin : **Pp0, PP0H, PP1, PP1H, PP2, PP2H PP3 PP3H, PP4, PP4H, PP5, PP5H, PP6, PP6H, PP7, PP7H, PP8, PP9H, GEN, Genh, Gen2, GEN2H, GEN3, GEN3H, GEN4, GEN4H, GPR, BPF, BPFH.**

5) Les entrées de la platine Amicus18

Jusqu’ici, avec notre platine Amicus, nous n’avons généré que des signaux de sortie. L’intérêt d’un microncontrôleur est de pouvoir gérer aussi les données en entrées.

Le système de développement Amicus18 dispose de deux types d'entrées : digitale (ou logique en couleur bleu) et analogiques (en couleur rouge) comme le montre l'image n°17. La différence provient du nombre maximal de valeurs que l'entrée peut lire. Les entrées analogiques sont marquées "ANx"où x est l'identification de l'entrée (0 à 4 et 8 à 12).



Entrées logiques (ou digitales)

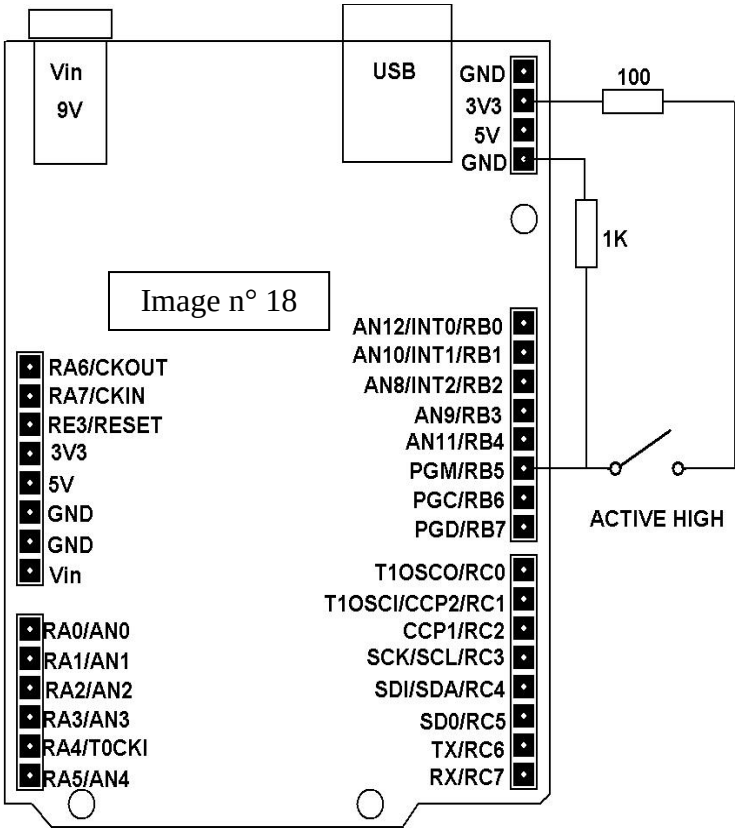
Tous les ports du système Amicus 18 peuvent être des entrées logiques. Le Port B (entrées RB0 à RB7) a même la possibilité d'activer des résistances "pull up" internes, ce qui simplifie l'implémentation des entrées digitales (dans le cas d'un raccordement à un clavier par exemple)

Dans le cas d'une entrée logique, seulement deux états sont possibles :

LOW (bas) = 0 V ou presque
HIGH (Haut) = 3,3V ou presque

Une entrée logique peut être "active high" ou "active low". Dans le cas où l'entrée est "active high", l'action est de lui appliquer une tension de +3,3 V lorsqu'elle est active (voir image n° 18). A ce moment, il faut reprendre sa casquette d'analogicien (car la loi d'Ohm continue de fonctionner) et faire attention à positionner l'entrée à un niveau bas (0 V) par une résistance montée entre l'entrée et la masse, pour prévenir des actions intempestives (effet de main par exemple).

D'autre part, la résistance de 100 ohms limite le courant vers la borne +3,3 V. Sinon, gare au court-circuit... Lorsque l'interrupteur est fermé, les deux résistances forment un pont diviseur et la tension sur le port sera proche de +3V (en non pas +3,3 V)



Dans le cas où l'entrée est "active low", il faut relier l'entrée à la masse. Pour stabiliser le circuit , une résistance "pull up" reliée au +3,3 V est nécessaire. (voir image n°19)

Dans tous les cas, une entrée digitale peut être actionnée par des transistors bipolaires, des FET, ou tout autre circuit logique. Attention à la tension d'entrée ! Le maximum à ne pas dépasser est 3,3 V. Sinon, vous pouvez dire adieu à votre Amicus...

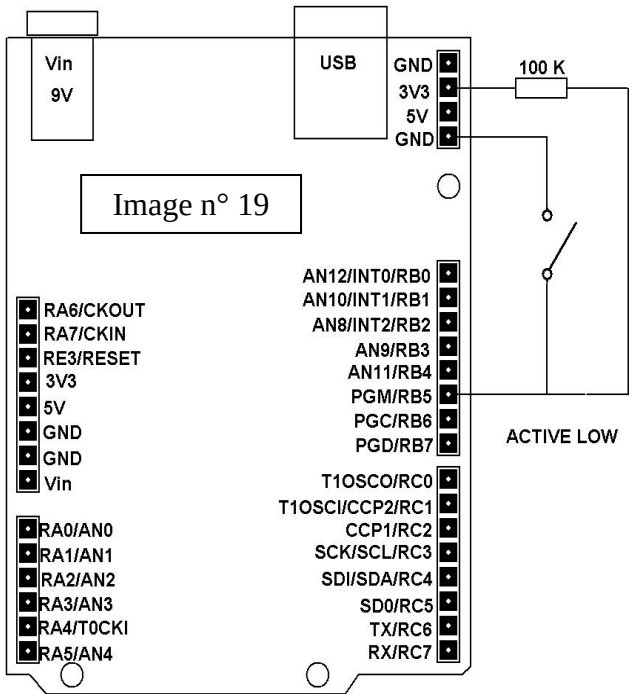


Image n° 19

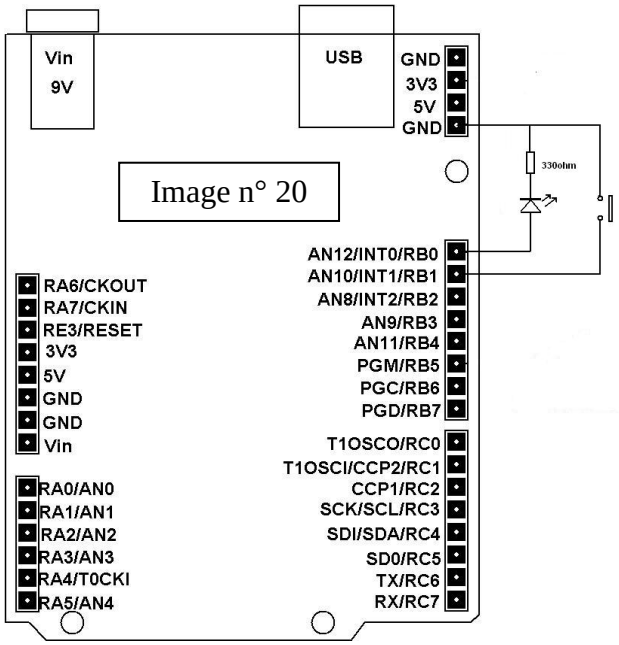


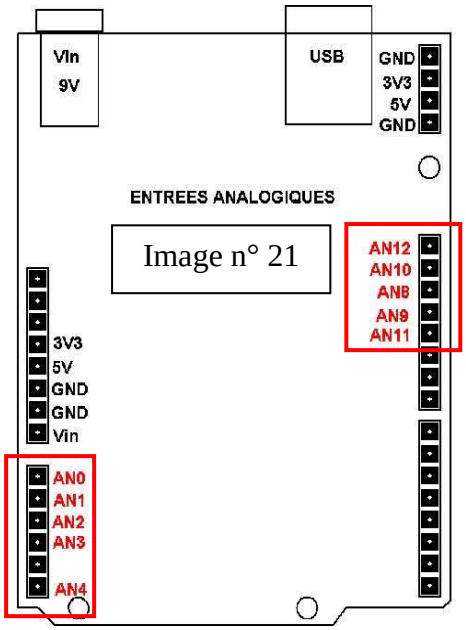
Image n° 20

Exemple de programme (image n° 20) :

```
'****************************************
'*  Name    : BOUTON_DEMO.BAS           *
'*  Author  : VladimirSTEPHAN           *
'*  Notice  : Copyright (c) 2010 F4FNA*
'*          : All Rights Reserved       *
'*  Date    : 10/12/2010                *
'*  Version : 1.0                       *
'*  Notes   : entrées/sorties PIC       *
'*          : demo action Bouton        *
'****************************************
Symbol bouton = PORTB.1  'bouton poussoir connecté entre le portB1 et la masse
Symbol LED = PORTB.0     'Diode LED entre le portB0 et la masse
Input  bouton            'Déclaration de l'entrée
PortB_Pullups = On       'Activation des résistances internes du PIC
debut:                   'Etiquette du début du programme
If bouton=1 Then         'Si l'entrée "bouton" est à 3.3V, alors le bouton n'est pas actionné, donc :
Low LED                  'La LED est éteinte (connectée entre la masse et le portB.0 qui est "LOW")
HRSOut "BOUTON OFF "     'on écrit sur le terminal RS232 (icône "Serial Com" du logiciel IDE) l'état du bouton (message "Bouton Off")
DelayMS 500              'on attend 500 ms (pour avoir le temps de lire le message sur le terminal)
GoTo debut               'on revient au début du programme pour un autre test du bouton
Else                     'Sinon, l'entrée est mise à la masse par le bouton
High LED                 'on allume la LED
HRSOut "BOUTON ON !!!"   'on écrit "BOUTON ON !!!" sur le terminal RS232 (icône "Serial Com" du logiciel IDE)
DelayMS 500              'on attend 500 ms (pour avoir le temps de lire le 'message sur terminal)
GoTo debut               'on revient au début du programme pour un autre test du bouton
EndIf                    'on referme le test du bouton
End                      'fin du programme (jamais atteint, car le programme est une boucle sans fin)
```

Entrées analogiques

Dans le cas d'une entrée analogique, le nombre des valeurs peut atteindre 1024 entre 0 V et +3,3 V, ce qui nous donne une précision de 3,22 mV (3,3 V / 1024). Les entrées analogiques disponibles sur Amicus18 sont : AN0 à AN4 sur le port A et AN8 à AN12 sur le port B (voir image n°21)



Pour pouvoir travailler avec les entrées analogiques, le PIC possède un convertisseur analogique-digital (ADC en anglais) et le compilateur dispose d'un ensemble de macrocommandes spécifiques (A/D Converter Macros) :

SetChanADC : sélectionne l'entrée analogique à utiliser
SelChanConvADC : sélectionne l'entrée analogique à utiliser et démarre la conversion A/D
OpenADC : configure le convertisseur analogique-digital
ReadADC : lit le résultat d'une conversion analogique-digitale.
ConvertADC : demarre une conversion analogique-digitale
BusyADC : regarde si une conversion A/D est en cours
CloseADC : arrête la conversion analogique-digitale

Exemple de programme (voir image n° 22 pour le schéma associé) :

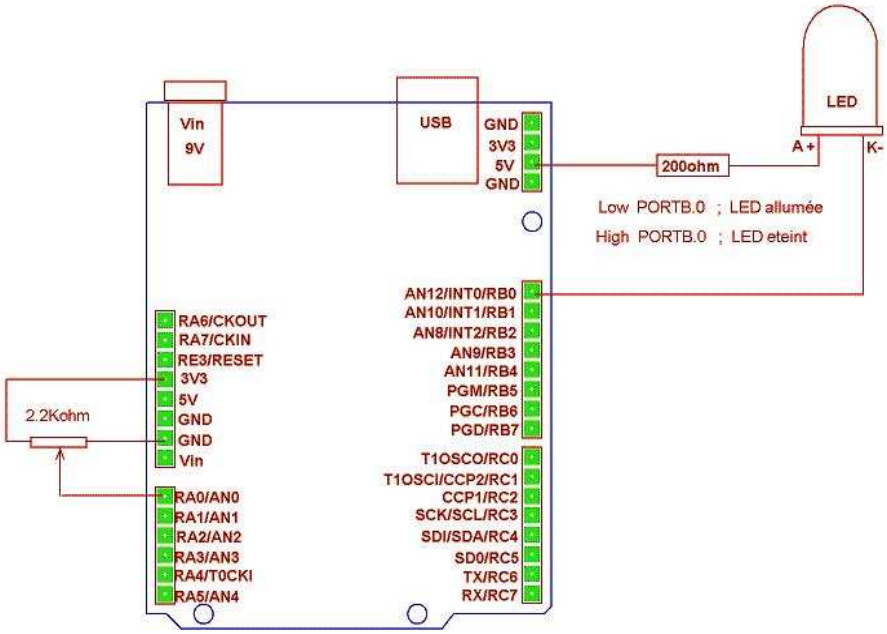
```
'*****************************************************************
'*  Name    : ADC_EXEMPLE.BAS                                    *
'*  Author  : VladimirSTEPHAN                                    *
'*  Notice  : Copyright (c) 2011 F4FNA                           *
'*          : All Rights Reserved                                *
'*  Date    : 21/01/2011                                         *
'*  Version : 1.0                                                *
'*  Notes   : Exemple conversion analogique, avec detection du   *
'*          : seuil de tension                                   *
'*****************************************************************
' Lire la valeur d'une tension sur l'entrée A0 et l'afficher
' sur le terminal série RS232 du logiciel IDE
' Compare la valeur lue avec un parametre préetabli et actionne une
' sortie pour signaler par l'extinction d'une LED le depassement de
' la valeur du paramètre
' Hardware:
' une LED connectée entre le +5v et l'entrée PORTB.0
' un potentiomètre avec une extremité vers la +3.3V, l'autre au GND
' et le coursseur a l'entrée analogique AN0
'-----------------------------------------------------------------------
    Include "ADC.inc"    ' Charge les macrocomandes specifiques.
    OpenADC(ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_2_TAD, ADC_REF_VDD_VSS, ADC_1ANA)
'   Ouvre le convertisseur analogique avec les caractéristiques suivantes:
'               Fosc/32
'               opération sur 10 bit justifié à droite
'               valeur du TAD = 2
'               Vref+ at Vcc : Vref- at Gnd
'               Make AN0 an analogue input
'-----Variables---------------------------------------------------------
    Dim ADC_Result As Word      ' Création d'une variable de 16bit non signée
                                ' pour les valeurs ADC
    Dim mV As Float             ' Création d'une variable à virgule flottante
                                ' pour afficher les résultats
    Dim offset As Float         ' Création d'une variable à virgule flottante
                                ' pour les calculs des valeurs
    Dim limite As Float         ' Création d'une variable à virgule flottante
```

```
                                ' pour le seuil de déclenchement
    offset=3.2                  ' 3300 mV /1024 des valeurs possibles = 3.2 mV
    limite =1.730               ' seuil de tension du déclenchement
'----------Programme-----------------------------------------------------------
    While 1 = 1                         'création d'une boucle sans fin
        ADC_Result = ADIn 0             'lire la valeur mesurée sur l'entrée
                                        'analogique
        mw= adc_result*offset+offset    'calcul de la valeur de la tension mesurée
        mw= mw/1000                     'transformation en volts
        HRSOut Dec mw," Volts", 13      'affiche sur le terminal série RS232
        If mw> limite Then              'comparaison: si le tension mesurée est
                                        'supérieure à la limite, alors:
        High PORTB.0                    'éteindre la LED
        Else                            'sinon
        Low PORTB.0                     'allumer la LED
        EndIf                           'fin de la comparaison
        DelayMS 1000                    'attendre 1 seconde
    Wend                                'reboucle a l'infini....
    End
```

Fonctionnement :

La LED s'allume si le seuil de tension est en dessous de la valeur du paramètre « limite » (1,73 V)

Lorsque la tension limite est dépassée, la diode s'éteint pendant au moins 1 seconde.



Notez la manière dont est structuré le programme :

- **l'entête** : nom du programme, son auteur, sa date de création puis le descriptif (tout ceci n'est pas « compilé »)
- **chargement** des commandes nécessaires au programme
- **déclaration** des variables : type de variables et valeurs. Si on veut changer une valeur (tension de déclenchement par exemple), il faut la modifier dans le programme puis le compiler à nouveau (le transformer dans un langage compréhensible par le microcontrôleur) et l'envoyer à la platine Amicus (icône Program du logiciel Amicus IDE)
- **le programme** en lui-même. Remarquez que les commentaires (tout ce qui commence par une apostrophe et qui est édité en bleu par le logiciel) ne seront pas compilés et ne seront pas transférés dans la mémoire de votre Amicus. Finalement, il ne reste que très peu d'instructions à compiler et à transférer dans la mémoire du microcontrôleur.

(Fin de la séance n°2)

Vladimir F4FNA *pour le Radio-Club de la Haute Île F5KFF/F6KGL.*

Rappel : les « Samedis Techniques » ont lieu tous les 2ème samedis de chaque mois de 14h00 à 17h00 dans les locaux du Radio-Club (Port de Plaisance, 93330 Neuilly sur Marne). Toutes les informations sur notre radio-club et sur ces réunions sont disponibles sur notre site : http://f6kgl/f5kff.free.fr . Tout le monde peut participer à ces réunions. N'hésitez pas à pousser la porte de notre radio-club : vous serez toujours les bienvenus !