AMICUS 18 (3ème partie)

Dans la seconde partie, nous avions évoqué l'utilitaire « Serial Terminal » mais nous n'avions pas vu ses fonctionnalités, nous allons commencer par les étudier puis nous reviendrons sur la programmation en langage BASIC en développant un chapitre entier sur les boucles et les tests conditionnels. Cet article est la synthèse de ce qui a été présenté par Vladimir F4FNA lors de la réunion du 11/02/11.

6) le « plug-in » Amicus Serial Terminal

Présent par défaut dans l'environnement de développement, le « plug-in » « **Amicus Serial Terminal** » est un outil très pratique pour tester et déboguer les programmes. Il est relié à travers la liaison USB, aux entrées/sorties sérielles du PIC (TX/RC6 et RX/RC7 en bas à droite de la platine, voir image n° 23) et on le fait fonctionner par les commandes BASIC HRSIn (Hardware RS232 In) et HRSOut (Hardware RS232 Out).

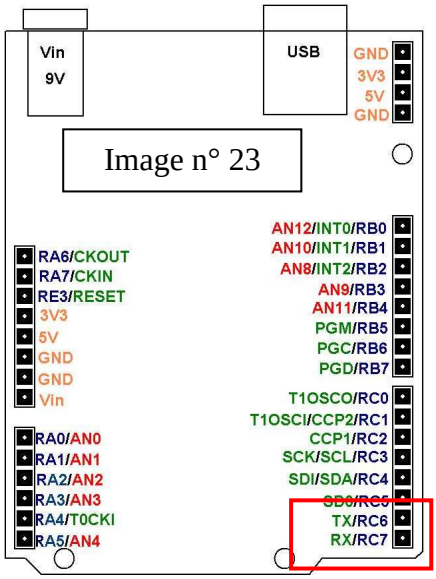

Image n° 23

Pour activer cette fonction, on doit tout d'abord **brancher la platine Amicus18 sur un port USB** du PC. Ensuite, on clique sur l'icône « **Serial Com** » située dans la barre de plug-ins. (voir image n° 24)

Image n° 24

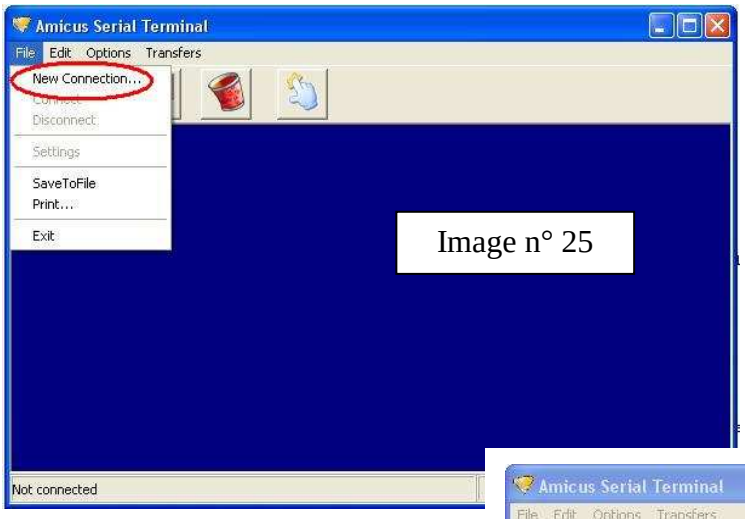

Image n° 25

Une fenêtre s'ouvre. On peut la « glisser-déplacer » où on le souhaite. Elle présente une barre de menus ressemblant à celle de l'application Hyperterminal de Windows (voir image n° 24).

En cliquant sur « **File** », un menu déroulant donne une seule possibilité de sélection : « **New Connection** » (voir image n° 25).

Dans la boite de dialogue (voir image n° 26), on cherche le **port COM** attribué à la platine Amicus18 par Windows puis on valide par **OK**.

Attention, cette opération n'est possible que si la platine Amicus18 est branchée sur un port USB du PC!

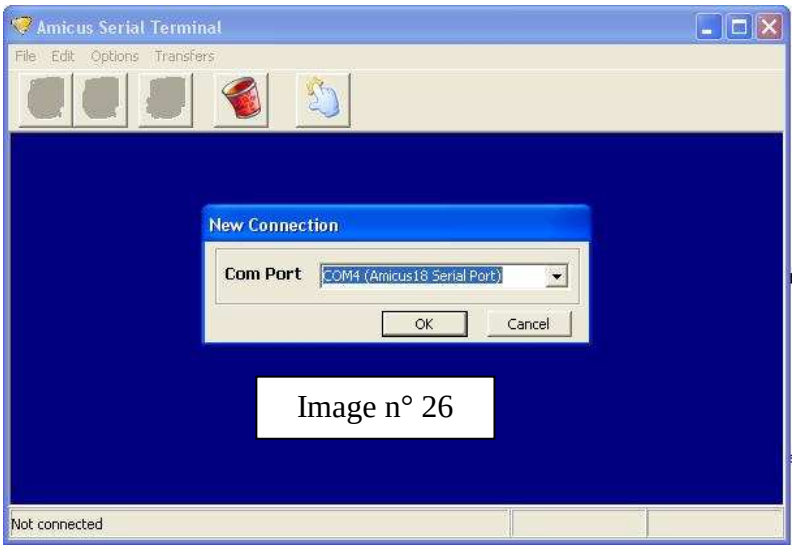

Image n° 26

On remarque les **icônes** visibles en dessous de la barre des menus en bas à gauche et l'inscription. Certaines sont grisées, d'autres changent d'aspect en fonction du contexte de l'utilisation. Les icônes affichées et leurs significations sont récapitulées ci-contre.

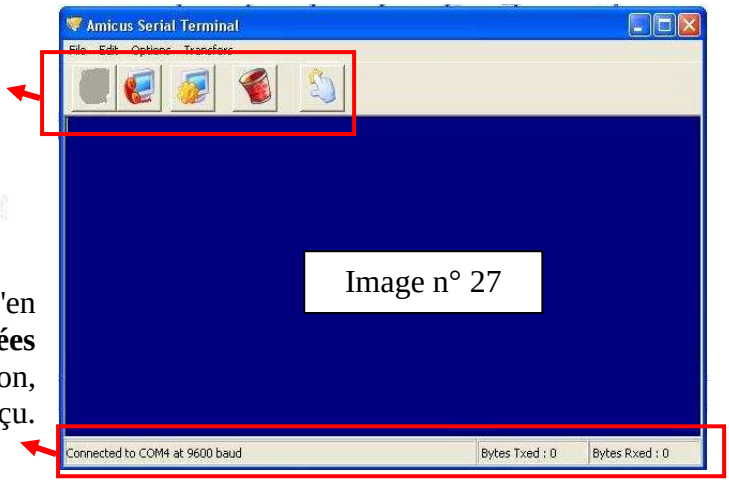

Image n° 27

En même temps, dans la zone de texte d'en bas de la fenêtre, sont affichées les **données** concernant la configuration de la connexion, ainsi que le nombre des Bytes émis et reçu. (voir image n° 27)

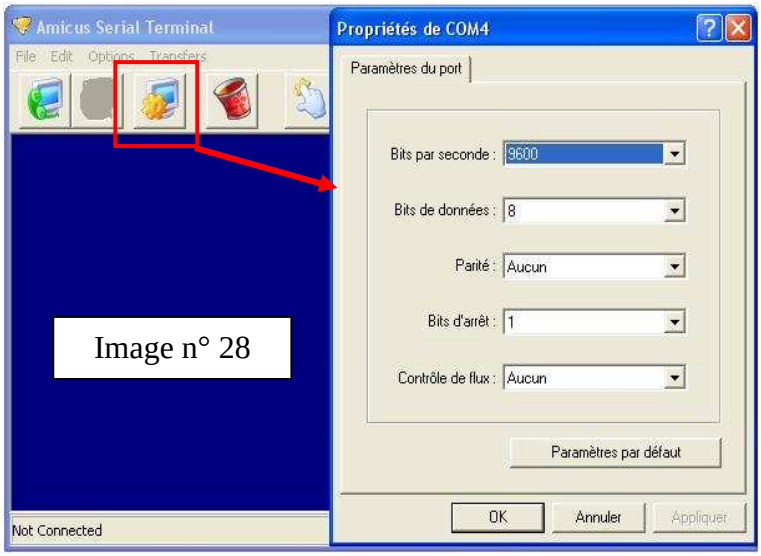

Image n° 28

Si on a besoin d'un changement des paramètres de la connexion du port série, on peut le faire en cliquant sur l'icône décrite plus haut (**configuration**), et ensuite remplir les cases dans la fenêtre de propriétés du port qui apparaît (voir image n° 28)

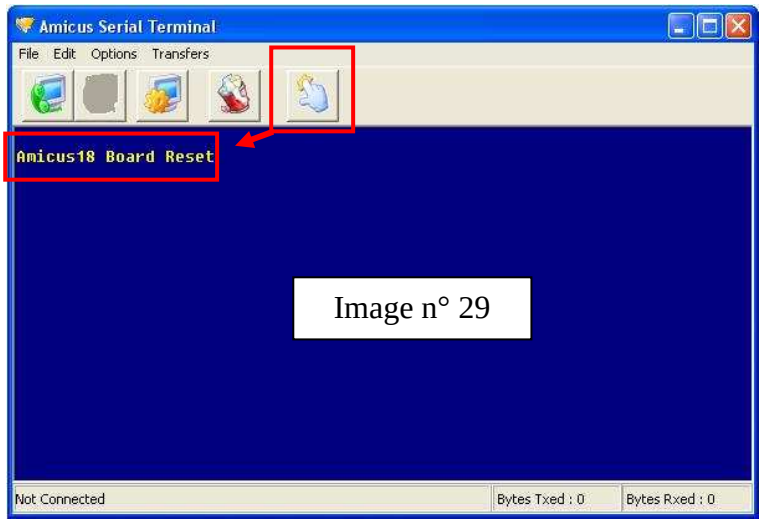

Image n° 29

On peut faire un **Reset** (Remise à zéro) de la platine Amicus18 à travers la liaison USB, en cliquant sur l'icône approprié (Reset). Cela nous évite d'appuyer sur le bouton poussoir « Reset » situé sur la platine. Un message nous confirme l'opération (voir image n° 29).

En fonctionnement normal, le terminal affiche les données envoyées par la platine Amicus18, (voir image n°30 pour la **restitution des données sur le « Serial Terminal** »). Dans cet exemple, le programme est conçu de telle manière que la diode restera éteinte tant que la tension est inférieure à la tension limite (1,730 volt). L'expérience a duré 16 secondes puisqu'on a mesuré 16 valeurs et que le programme mesure la tension toutes les secondes. La tension mesurée varie entre 1,974 et 1,977 volt. Cette tension étant supérieure à la

limite de déclenchement prévue dans le programme (1,730 volt), la diode restera éteinte tout le temps de la manipulation.

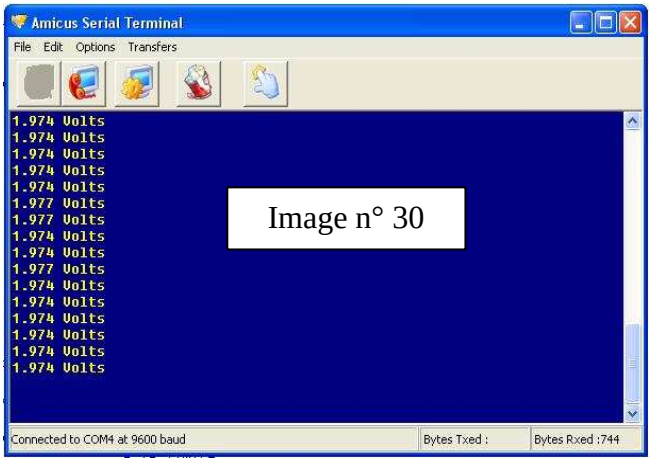

Image n° 30

```
'*****************************************
'*  Name    : UNTITLED.BAS                *
'*  Author  : VladimirSTEPHAN             *
'*  Notice  : Copyright (c) 2011 F4FNA    *
'*          : All Rights Reserved         *
'*  Date    : 21/01/2011                  *
'*  Version : 1.0                         *
'*  Notes   : Exemple conversion analogique*
'*          : avec détection du seuil de  *
'*          : tension                     *
'*****************************************
' Lire la valeur d'une tension sur l'entrée A0 et l'affiche sur le terminal
' série de l'IDE, compare la valeur lue avec un paramètre préétabli et actionne
' une sortie pour éteindre une LED lors du dépassement de la valeur du paramètre
' Hardware:
' une LED connectée entre le +5V et l'entrée PORTB.0
' un potentiomètre avec une extrémité vers le +3.3V, l'autre à la masse
' et le curseur à l'entrée analogique AN0
'-------------------------------------------------------------------
'   Include "ADC.inc"    ' Charge les macrocomandes spécifiques.
'   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 justifiés à droite
'               valeur du TAD = 2
'               Vref+ at Vcc : Vref- at Gnd
'               AN0 = entrée analogique
'-----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 = 3.2mV
    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
     mV= adc_result*offset+offset'calcul de la valeur de la tension mesurée
     mV= mV/1000                  'transformation en volts
     HRSOut Dec mV," Volts", 13   'affiche sur le terminal série
     If mV > limite Then          'comparaison : si la tension mesurée est
                                  'supérieure à la limite
     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 à l'infini....
```

7) programmation : les boucles et les tests

Revenons sur la programmation avec le compilateur BASIC de l'Amicus18 et arrêtons-nous sur les boucles et les tests en programmation. Nous les avions déjà un peu évoqué dans les parties précédentes avec la notion d'étiquette (« Label » en anglais).

Sauf dans des cas très rares, tout programme contient au moins une boucle.

- une boucle est une succession d'instructions où la fin coïncide avec le début.
- d'un point de vue temporel, une boucle peut être finie ou infinie
- la sortie de la boucle peut se faire avec une condition pour les boucles infinies ou à la fin des opérations pour les boucles finies

La fonction GoTo :

C'est le type de boucle le plus simple. Toutes les instructions comprises entre ETIQUETTE et GOTO ETIQUETTE sont exécutées sans fin.

exemple: LED qui clignote sans fin toutes les secondes

```
ETIQUETTE:                'ici commence la boucle
      DelayMS 500         'attente 500ms
      High LED            'La LED S'allume
      DelayMS 500         'attente 500ms
      Low LED             'la LED s'éteigne
      GoTo  ETIQUETTE     'on recommence sans fin
```

Pour que la boucle fonctionne sur une période de temps fini, on doit mettre en place une modification cyclique d'une variable et un test de condition de sortie de boucle (instruction If / Then étudiée plus loin).

Exemple :

```
  Symbol LED= PORTB.1     'la LED est branchée entre le portB1 et la masse
  Dim A As Byte           'déclaration d'une variable
  A=0                     'initialisation de la variable A
  ETIQUETTE:              'ici commence la boucle
      DelayMS 500         'attente 500ms
      High LED            'La LED s'allume
      DelayMS 500         'attente 500ms
      Low LED             'la LED s'éteind
      Inc A               'on augmente la valeur de "A" d'une unité
      If A=>10 Then       'Si la valeur de "A" est égale ou plus grande que 10
      GoTo APRES_BOUCLE   'Alors on sort de boucle (on va à « APRES_BOUCLE »)
      Else                'sinon
      GoTo  ETIQUETTE     'on recommence (on va à « ETIQUETTE »)
      EndIf               'fin du test (on n'y passe jamais mais le compilateur
                          'a besoin de cette instruction
APRES_BOUCLE:             'étiquette où on sort de la boucle
End                       'fin du programme
```

La fonction FOR / NEXT :

Le même principe est optimisé dans la boucle "FOR - NEXT" où la condition de sortie de la boucle est le comptage (ou le décomptage) avec une valeur de fin définie dans le programme. Une fois cette valeur atteinte, la sortie de la boucle est automatique et le programme reprend à l'instruction suivante (après la commande NEXT).

Exemple: la LED clignote 10 fois pendant 10 secondes

```
    Symbol Sortie_LED = PORTB.1'la LED est branchée entre le portB1 et la masse
    Dim i As Byte
      For i = 0 To 9      'pour 10 cycles
      DelayMS 500         'attente 500ms
      High LED            'La LED s'allume
      DelayMS 500         'attente 500ms
      Low LED             'la LED s'éteind
      Next                'on recommence si i<10
 End                      'fin du programme
```

La fonction REPEAT / UNTIL :

La boucle "REPEAT – UNTIL" (Répète – jusqu'à) est une boucle où la condition de fin de boucle n'est pas forcément la valeur d'un compteur mais n'importe quelle valeur d'une variable. Toutes les

instructions comprises entre «REPEAT » et « UNTIL » sont exécutées tant que la condition de sortie n'est pas respectée : on répète les instructions jusqu'à ce que la condition soit remplie.

Exemple: LED clignotante avec sortie sur appui sur un bouton poussoir branché entre le port B0 de l'Amicus18 et la masse :

```
      Symbol LED= PORTB.1      'la LED est branchée entre le portB1 et la masse
      Symbol BOUTON= PORTB.0   'Le bouton est branché entre le portB0 et la masse
      PortB_Pullups=TRUE
      Repeat                   'ici commence la boucle
      DelayMS 500              'attente 500ms
      High LED                 'La LED s'allume
      DelayMS 500              'attente 500ms
      Low LED                  'la LED s'éteind
      Until PORTB.0 = 0        'condition de sortie de la boucle (contact entre
                               'le port B0 et la masse)
End                            'fin du programme
```

La fonction WHILE / WEND :

La boucle "WHILE – WEND" fonctionne d'une manière inverse : elle se reboucle seulement si la condition de fonctionnement est respectée (While signifie « Tant que » en anglais). Toutes les instructions comprises entre le «WHILE » et le « WEND » sont exécutées tant que la condition de fonctionnement de la boucle est respectée.

Les sauts (GoTo et GoSub / Return)

On a vu un peu plus haut que pour faire une boucle, on doit faire un saut de la fin vers le début (on revient en arrière). Les sauts nous permettent d'atteindre d'autres endroits du programme identifiés par une étiquette.

Dans le compilateur Amicus18, on distingue deux types de sauts : « GoTo » et « GoSub ». La différence entre les deux est que « GoTo » indique au programme d'aller vers une étiquette puis de continuer à partir de cette étiquette tandis que « GoSub » indique au programme d'aller vers une « sous-routine » qui exécute les opérations jusqu'à l'instruction « Return » puis retourne au programme principal et exécute l'instruction suivante.

GoTo sert principalement pour atteindre des parties de programme suite à un test de valeur d'une variable (voir plus haut les exemples de boucle avec test de condition de sortie)

GoSub aide à écrire un code plus lisible et évite les répétitions en transformant les tâches répétitives en « sous-routines » logées dans le programme, soit au début après la description des variables (exemple 1), soit après l'instruction **End** indiquant la fin du programme principal (exemple 2).

Exemple 1 : les sous-routines sont au début du programme :

```
'*************************************************************
'*  Name    : audio_logic_tester.BAS                         *
'*  Author  : Vladimir STEPHAN                               *
'*  Notice  : Copyright (c) 2011 F4FNA                       *
'*          : All Rights Reserved                            *
'*  Date    : 14/02/2011                                     *
'*  Version : 1.0                                            *
'*  Notes   : demo sous-routines                             *
'*          :                                                *
'*************************************************************
' indicateur sonore d'état logique cmos.
' low = 0 à 0,8V     son 800Hz
' high = 2,7 à 3,3V  son 2.7 KHz
' pour 0,9 à 2,69V   pas de son
    Include "ADC.inc"    ' Charge les macro-commandes spécifiques.
    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 bits justifiés a droite
'               valeur du TAD = 2
'               Vref+ at Vcc : Vref- at Gnd
'               Make AN0 an analogue input
    Symbol son_out=PORTC.0
'-----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 = 3,2mV
    limite =1.730               ' seuil de tension du déclenchement
'----------Sous-routines---------------------------------------------------
 GoTo apressousroutines         'indique au programme de passer directement
                                'au programme principal
son_low:                        'sous-routine appelée pour 0 à 0,8V
 FreqOut son_out, 800,10        'émission d'un son de 800Hz pour 10mS
Return                          'retour au programme principal
son_high:                       'sous-routine appelée pour 2,7 à 3,3V
FreqOut son_out, 2700,10        'émission d'un son de 2700Hz pour 10mS
Return                          'retour au programme principal
'----------Programme-------------------------------------------------------
apressousroutines:
  While 1 = 1                       'tant que 1 = 1 : c'est une boucle sans fin
     ADC_Result = ADIn 0            'lire la valeur mesurée sur l'entrée
                                    'analogique RA0/AN0 de l'Amicus18
     mV= adc_result*offset+offset   'calcul de la valeur de la tension mesurée
     mV= mV/1000                    'transformation en volts
     Select ADC_Result              'test du niveau logique
     Case 0 To 0.8                  'si le niveau est compris entre 0 et 0,8V
     GoSub son_low                  'va exécuter la sous-routine son_low
     Case 2.7 To 3.3                'si le niveau est compris entre 2,7 et 3,3V
     GoSub son_high                 'va exécuter la sous-routine son_high
     Case Else                      'dans tous les autres cas,
     EndSelect                      'finit le test conditionnel du niveau
  Wend                              'reboucle à l'infini (en revenant à While)
End                                 'fin du programme principal (jamais atteint)
```

Dans ce programme, les sous-routines sont comprises entre la ligne « **GoTo** apressousroutines » et la ligne « apressousroutines: » où le saut **GoTo** nous évite une utilisation hasardeuse des sous-routines au début du programme. De ce fait, les sous-routines sont utilisables seulement par la commande « **GoSub** <étiquette> ». Les commandes « **Select / EndSelect** » et « **Case / Case Else** » sont étudiées plus loin dans les tests conditionnels.

Exemple 2 : les sous-routines sont après l'instruction « End » (fin du programme principal).

La définition des variables et des macro-commandes n'a pas été reprise (identique à l'exemple 1).

```
'----------Programme-------------------------------------------------------
  While 1 = 1                   'création d'une boucle sans fin
    ADC_Result = ADIn 0         'lire la valeur mesurée sur l'entrée analogique
    mV= adc_result*offset+offset'calcul de la valeur de la tension mesurée
    mV= mV/1000                 'transformation en volts
    Select ADC_Result           'test du niveau logique
     Case 0 To 0.8              'si le niveau est compris entre 0 et 0,8V
     GoSub son_low              'va exécuter la sous-routine son_low
     Case 2.7 To 3.3            'si le niveau est compris entre 2,7 et 3,3V
       GoSub son_high           'va exécuter la sous-routine son_high
       Case Else                'dans tous les autres cas,
```

```
      EndSelect                  'finit le test conditionnel du niveau
    Wend                         'reboucle à l'infini....
End                              'fin du programme principal (jamais atteint)

'----------Sous-routines-------------------------------------------------------
son_low:                        'sous-routine appelée pour 0 à 0,8V
 FreqOut son_out, 800,10        'émission d'un son de 800Hz pour 10mS
Return                          'retour au programme principal
son_high:                       'sous-routine appelée pour 2,7 à 3,3V
FreqOut son_out, 2700,10        'émission d'un son de 2700Hz pour 10mS
Return                          'retour au programme principal
```

Les tests conditionnels (“If / Then” et “Select / Case”)

Lorsqu’on doit faire des tests de condition pour pouvoir prendre des décisions, selon les valeurs mesurées, on utilise des tests conditionnels. Le compilateur dispose de deux types de tests :

Le test “if-then-else-endif”

Le classique « If / Then » (si / alors) permet de tester une valeur et d’exécuter soit la première commande lorsque la condition est remplie soit la seconde commande (entre « else » et « endif »).

Exemple :

```
Etiquette :
If A=>10 Then        'Si la valeur de "A" est égale ou plus grande que 10
GoTo APRES_BOUCLE    'Alors on sort de la boucle
Else                 'sinon
GoTo  ETIQUETTE      'on recommence
EndIf                'fin du test
Apres_boucle :
```

On peut imbriquer plusieurs tests à condition de prendre soin de bien sortir des séquences de tests dans le bon ordre (il y a autant de « if » que de « then », « else » et « endif ») :

```
test_1 :      if<condition_1>then<execute1>else
test_2 :            if<condition_2>then<execute2>else
test_3 :                  if<condition_3>then<execute3>else<execute4>
fintest_3                 endif
fintest_2           endif
fintest_1     endif
```

Le test “select – case – case else – endselect”

Ce deuxième type de test conditionnel est plus gourmand en mémoire. En revanche, il est plus puissant et plus compréhensible dans un programme. Nous avons utilisé cette commande plus haut dans les exemples de sous-routines.

Exemple :

```
      Select ADC_Result          'test du niveau logique
      Case 0 To 0.8              'si le niveau est compris entre 0 et 0,8V
      GoSub son_low              'va exécuter la sous-routine son_low
      Case 2.7 To 3.3            'si le niveau est compris entre 2,7 et 3,3V
      GoSub son_high             'va exécuter la sous-routine son_high
      Case Else                  'dans tous les autres cas,
      EndSelect                  'fin du test conditionnel du niveau
```

(Fin de la séance n° 3)

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 !