Cours 02
Instructions étudiées

Configuration

- **$Baud**
- **Dim**

Programmation

- **Print**
- **Incr / Decr**
- **+ / -**
- **If Then Else**
  End If…
- **Symboles**

$Baud

Demande au compilateur de réécrire le baud rate réglé dans l'option menu.

Le baud rate, qui sera utilisé, doit être une constante numérique. Le baud rate peut être choisi dans le réglage du compilateur. Il est écrit dans un fichier de configuration. Dans le rapport, on peut voir quel baud a été utilisé. Il dépend aussi du quartz et du microprocesseur. Le simulateur ne tient pas compte des éventuelles erreurs de choix. Un mauvais réglage peut donner des résultats faux. Pour de bons résultats, il est préférable d'utiliser un quartz dont la fréquence est un multiple du baud rate. Certains microcontrôleurs AVR ont 2 UART. Par exemple: le Mega161, Mega162, Mega103 et le Mega128. Certains en ont même 4 UARTS.

```
'//////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 2
'
'  Revision 1.00
'
'  _____________
'  Ajout de $Baud
'
' 1200 – 2400 – 4800 – 9600 – 19200 – 28800 – 38400 – 57600 - 115200
'
'//////////////////////////////////////////////////////////////////


   $Baud = 9600



'Fin de la configuration du Micro
'_____________
```

Print

Envoie des données à la sortie RS232.

PRINT Var ; ["constant"] La variable ou constante à envoyer

On peut utiliser une virgule pour séparer des données différentes. Pour visualiser l'effet de PRINT, on peut utiliser le TERMINAL EMULATOR.

```
'////////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 2
'
'  Revision 1.00
'
'  _____________
'  Ajout de Print
'
'////////////////////////////////////////////////////////////////////


   Print "Vous avez presse sur le bouton..." , bnbrpressbutton




'Fin de la configuration du Micro
'_____________
```

Decr / Incr

Décrémente (incrémente) une variable d'une unité.

Syntaxe

Decr var / Incr var

Variable à décrémenter / incrémenter, (n'importe quelle variable numérique sauf BIT).

```
'////////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 2
'
'  Revision 1.00
'
'  _____________
'  Ajout de Print
'
'////////////////////////////////////////////////////////////////////

 Dim bVariable as Byte

   Incr bVariable

   Decr bVariable

'Fin de la configuration du Micro
'_____________
```

+ / -

Décrémente (incrémente) une variable d'une unité.

Syntaxe

Var = var + 1 / var =var - 1

Variable à décrémenter / incrémenter, (n'importe quelle variable numérique sauf BIT)

```
'////////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 2
'
'  Revision 1.00
'
'  _____________
'  Ajout de Print
'
'////////////////////////////////////////////////////////////////////

 Dim bVariable as Byte

   bVariable = bVariable + 1

   bVariable = bVariable - 1


'Fin de la configuration du Micro
'_____________
```

Dim

Qu'est-ce qu'une variable ?

Une variable est un objet défini par le programmeur. Cet objet numérique (même les mots sont des valeurs numériques!) change d'état pendant le déroulement du programme.

Un bouton, qui change d'état, fait varier un port d'entrée de 0 à 1. On va associer ce bouton à une variable et déclencher un process. (par exemple la fermeture d'un relais).

```
Dim Boutonstop as bit rem déclare la variable Boutonstop en tant que bit

Do                          rem boucle do loop until (faire tourner jusqu'à)
        Boutonstop=pind.5 rem pind.5 est la 5° patte du port D
loop until Boutonstop=0  rem boucle jusqu'à ce que pind.5 soit mise à 0
```

Dim

Les types de variables:

Attention le mot « Type » est une notion importante; on doit **TYPER** les variables avec les instructions **Dim**, **Local et Function.**

- **BIT** : dans l'exemple ci-dessus, la boucle attend que le boutonstop soit basculé; la variable peut prendre 2 états : 1 et 0.
- **BYTE** : Avec 8 bits, on compte jusqu'à &HFF soit 255 la variable associée est du type Byte.
- **INTEGER, WORD** : Avec 16 bits, on compte jusqu'à &HFFFF soit 65535. Suivant l'utilisation, on a les variables signées du type **Integer** de –32768 à+32767 ou du type **Word** de 0 à 65535.
- **LONG** : Avec 32 bits (&HFFFFFFFF), on compte de –2.147.483.648 à +2.147.483.647 et on aura les **long** (correspondant aux Integer).

Dim

Les types de variables:

Attention le mot « Type » est une notion importante on doit **TYPER** les variables avec les instructions **Dim**, **Local et Function.**

- **SINGLE** : ces variables signées et stockées sur 32 bits, permettent de calculer de 1.5X10-45 à3.4 x 1038 mais la quantité de chiffres significatifs est limitée ! Les variables de types Single permettent de travailler en virgules flottantes (voir opérateurs mathématiques).
- **DOUBLE** : Variables signées sur 64 bits comme les Single mais de –E308 à E308- nécessite l'implémentation de la librairie double.lbx fourni.
- **STRING** : Ce sont les lettres, mots et phrases jusqu'à 254 bytes. Leurs longueurs sont déclarées : ex : Discours as string * 10. (les strings occupent 1 byte de plus que leur déclaration.) Voir chapitre "Conversions de variables"

Dim

Les types de variables:

Attention le mot « Type » est une notion importante, on doit **TYPER** les variables avec les instructions **Dim**, **Local et Function.**

• **TABLEAUX (ARRAYS):** ce sont des groupes de variables indexées. Elles regroupent sous un même nom des variables qui seront appelées par un index. Elles sont déclarées par un nom suivi par le nombre d'index entre ( ). Le nombre d'éléments(index) doit être inférieur ou égal à 65535. Attention! l'index 0 n'existe pas. (voir le programme Array.bas) (contrairement au QuickBasic). Par exemple, la température en fonction de l'heure :

```
Dim temperature(24) as integer ' les températures peuvent être négatives
Dim J as byte
Temperature(1)=10
Temperature(2)=15
Temperature(3)=portA
Temperature(J)=20
```

Nom variable

- La déclaration des variables est obligatoire.

Dim déclare les variables globales du programme principal, **Local** les variables des SUB et des fonctions.

Le nom peut avoir de 1 à 32 caractères. Le premier caractère doit être une lettre, l'usage des mots réservés n'est pas autorisé sauf quand ils sont incorporés dans le nom de variable : *oper**AND**e* est accepté; donc utiliser le français, mais sans accent ! Pour mes variables génériques, celles qui servent dans les boucles, j'utilise J,K,Y et i (En BASCOM pour cette dernière). C'est une mauvaise habitude car I (i) = I (L)). Il n'y a pas de différence entre les minuscules et les majuscules : PinStop = pinstop. Dans les dernières versions de Bascom, les variables sont automatiquement réécrites avec une majuscule en entête, suivies de minuscules – voir option dans environnement.

Certains BASIC utilisent des suffixes pour caractériser les variables : $,%, &.

Dans le Basic Bascom, il vaut mieux éviter. Par exemple : mot$, permet seulement de se souvenir que mot$ est une variable **string**. Varinteger% n'est pas autorisé, il est utilisé dans les déclarations de data.

Questions

1) La température de fusion d'un alliage est de 1227°. Quel va-t-être le type de variable utilisé pour les calculs?

A String

B Single

C Integer

2) L'eau de mer gèle à –9°. Quel va-t-être le type de variable utilisé pour les calculs ?

A byte

B Word

C Integer

3) L'eau distillée gèle exactement à 0.0°C. Quel va-t-être le type de variable utilisé pour les calculs ?

A Single

B Word

C Integer

Questions

4) Un seul nom de variable est correct pour chaque proposition. Expliquer pourquoi.

A 0tello

B Otéllo

C Otello

5) Température consomme plus de Sram que T (ces deux variables ayant le même type).

A Non, c'est pareil.

B Température est plus gourmande.

C Température n'est pas un nom de variable accepté.

Réponse:

Question 1 : C = integer
Question 2 : C = integer
Question 3 : A = single
Question 4 : C = Otello
Question 5 : A = c'est pareil

IF..then..Elseif...then...Else...end If

On peut aussi écrire cette instruction sur une seule ligne quand elle ne comporte qu'un choix :

If pression>200 Then Stopmachine=0 End If

Les choix suivants sont facultatifs. **Elseif** doit être suivi d'une autre condition et de **then Else.**

```
  $regfile = "m328pdef.dat"              'Assigne le DEF du Micro

  $HwStack = 48
  $SwStack = 32
  $FrameSize = 32
  $Crystal = 16000000

  $Baud = 9600

'Fin de la configuration du Micro
'-----------------

Dim bValeur as Byte


    Do

        If bValeur = 5 Then
          'Faire quelque chise
        Else
          'Faire autre chose
        End If

    Loop


End
```

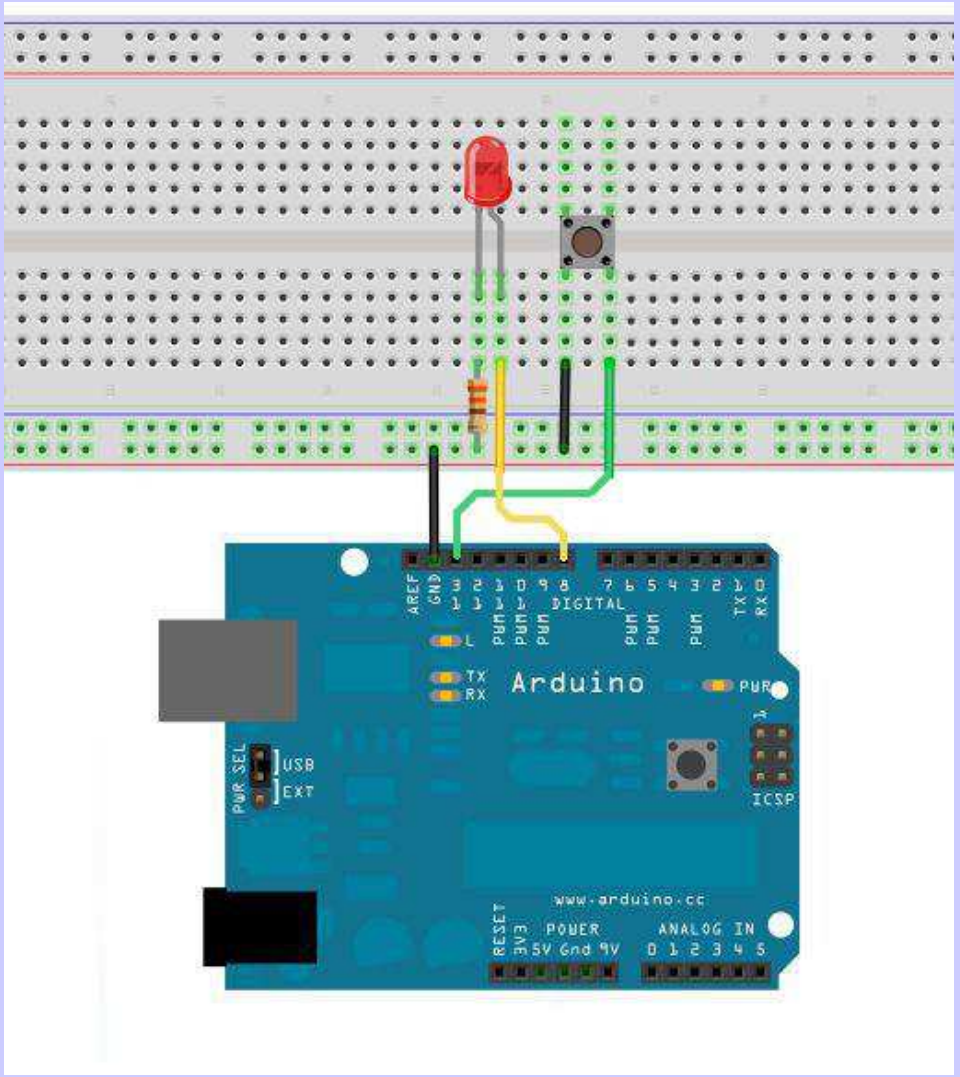
If Then...

```
  $regfile = "m328pdef.dat"              'Assigne le DEF du Micro

  $HwStack = 48
  $SwStack = 32
  $FrameSize = 32
  $Crystal = 16000000

  $Baud = 9600

'Fin de la configuration du Micro
'-----------------
'Fin de la configuration du Micro
'-----------------

Dim bValeur as Byte



    Do

        If bValeur = 5 Then
          'Faire quelque chise
        ElseIf bValeur = 8 Then
          'Faire autre chose
        ElseIf bValeur > 10 Then
          'Faire autre chose
        Else
          'Faire autre chose
        End If

    Loop


End
```

Exercise #4

Faire changer l'état d'un LED avec un interrupteur et envoyer des DATAS sur le port série. Faire aussi le suivi avec une variable qui sera, elle aussi, envoyée par le port série..

TIPS : Doit avoir une double vérification de l'entrée car un contact mécanique fait des rebonds.



Aide #4

Nouvelles commandes à utiliser :

Configuration :
$Baud
Debounce
Dim

Code :
Incr
Print
[Label]

Code #4

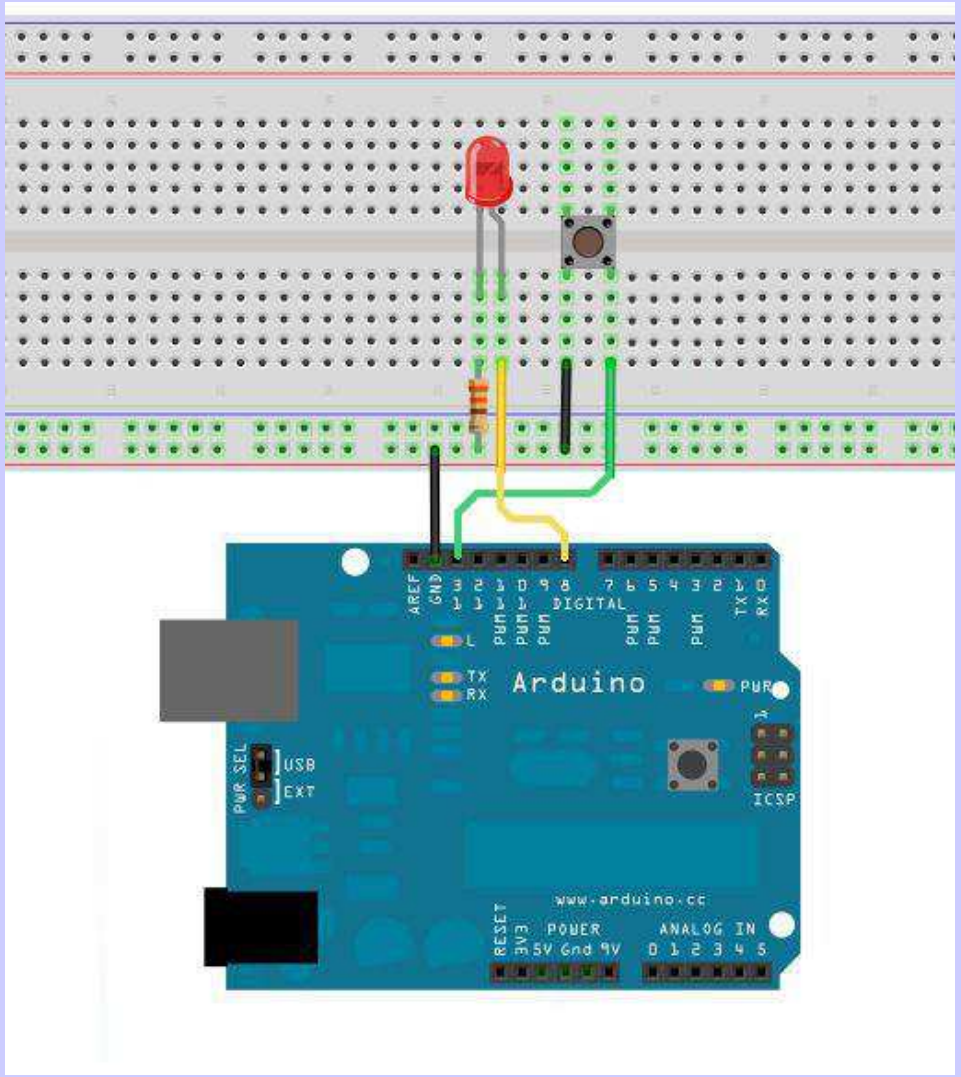
```
'////////////////////////////////////////////////////////////////////
'
' By Steve Tremblay
'
' Cours # 2
'
' Revision 1.00
'
' ____________
'
' FLASH 1 LED avec SW et envoie sur le port série
'
'////////////////////////////////////////////////////////////////////


  $regfile = "m328pdef.dat"                'Assigne le DEF du Micro

  $HwStack = 48
  $SwStack = 32
  $FrameSize = 32
  $Crystal = 16000000

  $Baud = 9600

  Dim bNbrPressButton as Byte

  Ddrb = &B00000001          'Config le PORT B   0 = Input / 1 = Output
                    'Assigne les PULL UP
  Portb.0 = 1               'Arduino # pin -> 8
  Portb.1 = 1               'Arduino # pin -> 9
  Portb.2 = 1               'Arduino # pin -> 10
  Portb.3 = 1               'Arduino # pin -> 11
  Portb.4 = 1               'Arduino # pin -> 12
  Portb.5 = 1               'Arduino # pin -> 13
  Portb.6 = 1               'Existe pas
  Portb.7 = 1               'Existe pas

  Config Pinb.5 = Input

'Fin de la configuration du Micro
```

```
 Do

      Debounce pinb.5 , 0 , Switch , Sub

    Loop


End



'_____________________________________

Switch:
  Toggle Portb.0
  Incr bNbrPressButton
  Print "Vous avez presser sur le bouton..." ; bNbrPressButton

Return
```

Exercise #5

Faire changer l'état d'un LED avec une interrupteur et envoyer des DATAS sur le port série. Faire des bonds de 5 – 8 – 12, et faire le suivi avec une variable qui sera, elle aussi, envoyée par le port série..

TIPS : Faire addition et envoyer la valeur.



Aide #5

Nouvelle commande à utiliser :

Configuration :

Code :
Incr ou +

Code #5

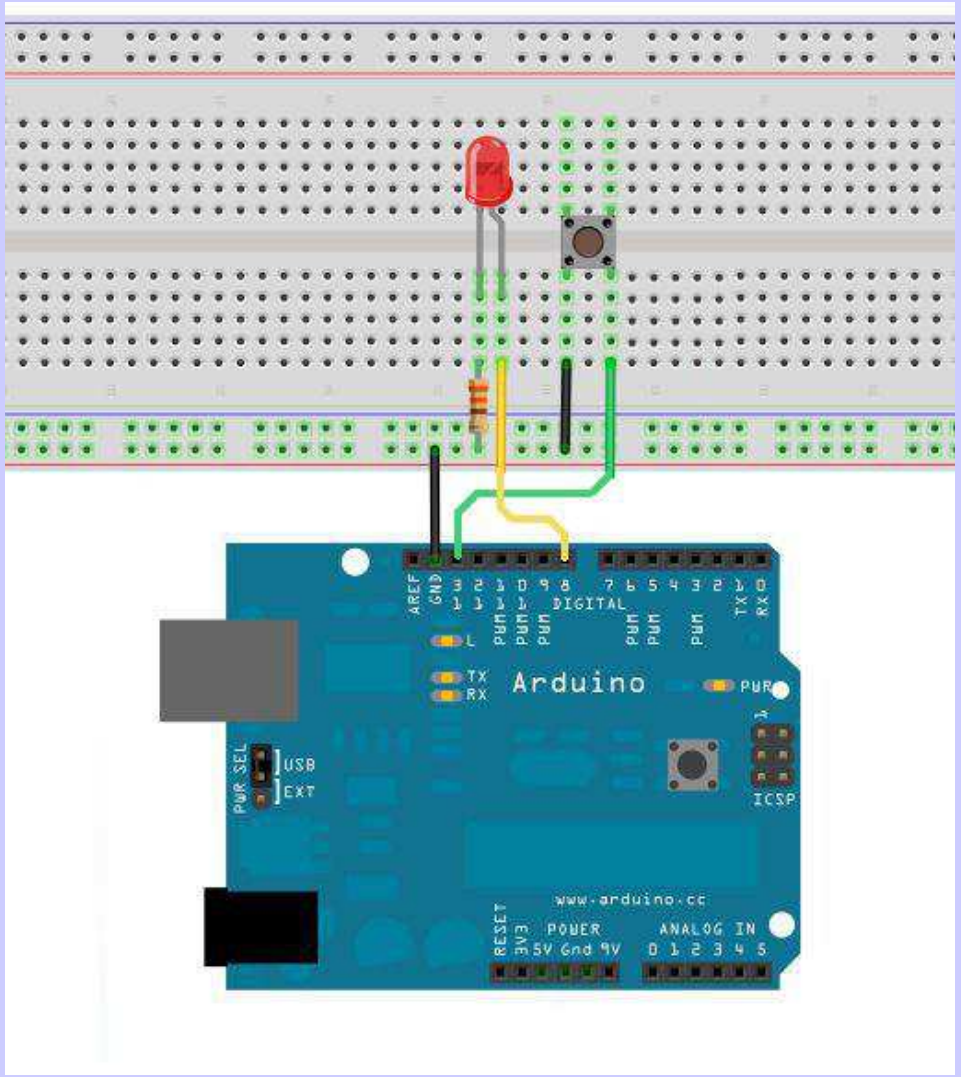
```
'////////////////////////////////////////////////////////////////
'
' By Steve Tremblay
'
' Cours # 2
'
' Revision 1.00
'
' ____________
'
' FLASH 1 LEDS avec SW et gestion dans LOOP
'
'////////////////////////////////////////////////////////////////


  $regfile = "m328pdef.dat"                'Assigne le DEF du Micro

  $HwStack = 48
  $SwStack = 32
  $FrameSize = 32
  $Crystal = 16000000

  $Baud = 9600

  Ddrb = &B00000001          'Config le PORT B   0 = Input / 1 = Output
                    'Assigne les PULL UP
  Portb.0 = 1               'Arduino # pin -> 8
  Portb.1 = 1               'Arduino # pin -> 9
  Portb.2 = 1               'Arduino # pin -> 10
  Portb.3 = 1               'Arduino # pin -> 11
  Portb.4 = 1               'Arduino # pin -> 12
  Portb.5 = 1               'Arduino # pin -> 13
  Portb.6 = 1               'Existe pas
  Portb.7 = 1               'Existe pas


Dim bNbrPressButton as byte
Dim bBoutonEvent as Byte

'Fin de la configuration du Micro
```

```
Print "Demarrage de l'application en cours...."


    Do

      Debounce pinb.5 , 0 , Switch , Sub

        If bBoutonEvent = 1 Then
          bBoutonEvent = 0
            If bNbrPressButton = 5 Then
              Print "Vous avez presser sur le bouton et la variable a une valeur de
5..."
            ElseIf bNbrPressButton = 8 Then
              Print "Vous avez presser sur le bouton et la variable a une valeur de
8..."
            ElseIf bNbrPressButton = 12 Then
              Print "Vous avez presser sur le bouton et la variable a une valeur de
12..."
              bNbrPressButton = 0
            End If
        End If

    Loop


End



'_____________________________________

Switch:

'  Incr bNbrPressButton
  bNbrPressButton = bNbrPressButton + 1
  bBoutonEvent = 1
  Toggle portb.0


Return
```

Exercise #6

Faire changer l'état d'un LED avec un interrupteur et envoyer des DATAS sur le port série. Faire des bonds de 5 – 8 – 12 et faire le suivi avec une variable qui sera, elle aussi, envoyée par le port série. Reporter le code dans la boucle principale.

TIPS : Mettre événement dans une variable.



Code #6

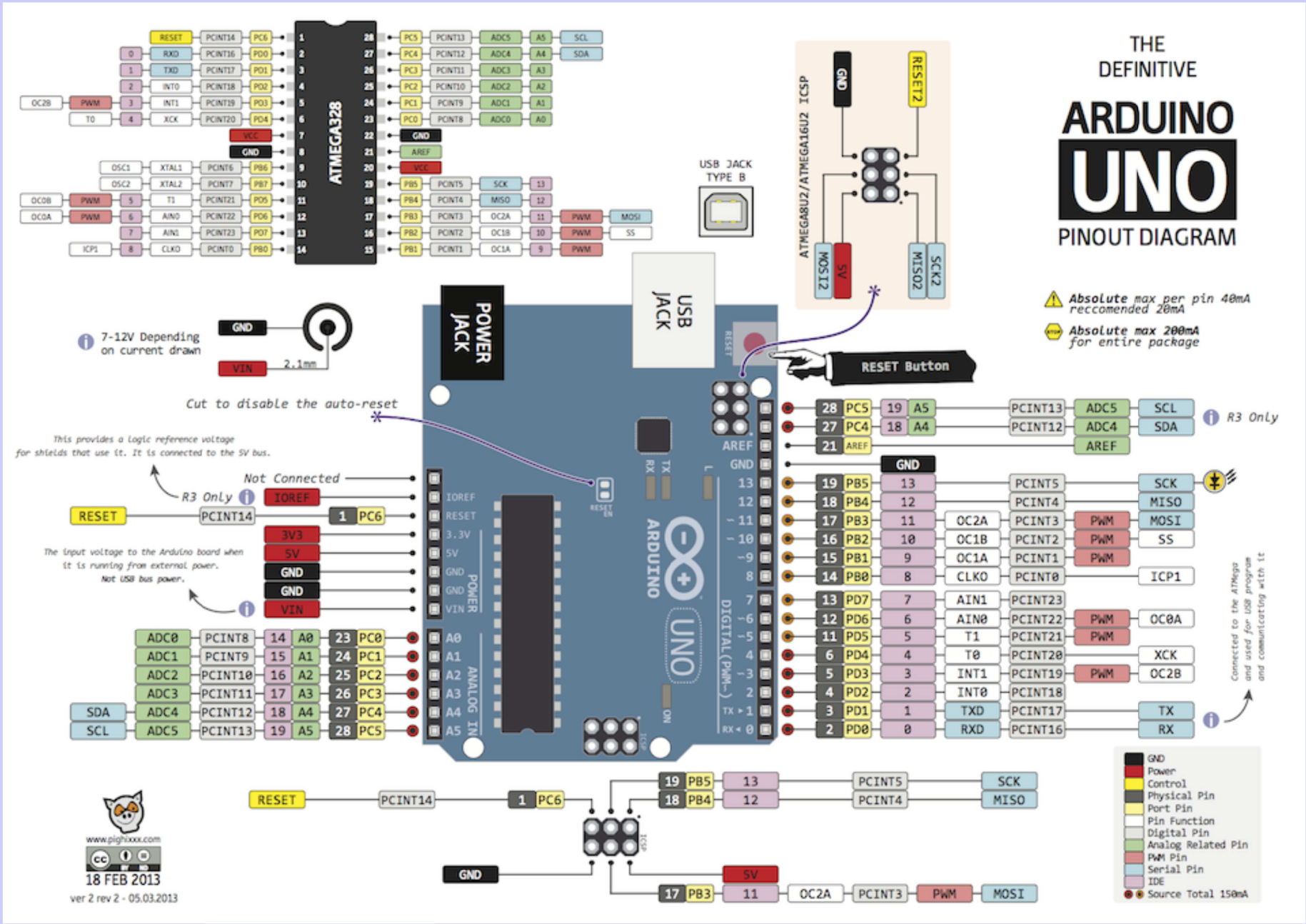
```
'////////////////////////////////////////////////////////
'
'   By Steve Tremblay
'
'   Cours # 1
'
'   Revision 1.00
'
'   ---------------------------
'
'   FLASH 1 LEDS avec SW et gestion dans LOOP
'
'////////////////////////////////////////////////////////

   $regfile = "m328pdef.dat"        'Assigne le DEF du Micro

   $HwStack = 48
   $SwStack = 32
   $FrameSize = 32
   $Crystal = 16000000

   $Baud = 9600

   Ddrb = &B00000001                'Config le PORT B   0 =
Input / 1 = Output
                                    'Assigne les PULL UP
   Portb.0 = 1                      'Arduino # pin -> 8
   Portb.1 = 1                      'Arduino # pin -> 9
   Portb.2 = 1                      'Arduino # pin -> 10
   Portb.3 = 1                      'Arduino # pin -> 11
   Portb.4 = 1                      'Arduino # pin -> 12
   Portb.5 = 1                      'Arduino # pin -> 13
   Portb.6 = 1                      'Existe pas
   Portb.7 = 1                      'Existe pas


Dim bNbrPressButton as byte
Dim bBoutonEvent as Byte

'Fin de la configuration du Micro
```

```
   Print "Demarrage de l'application en cours...."


      Do

         Debounce pinb.5 , 0 , Switch , Sub

            If bBoutonEvent = 1 Then
               bBoutonEvent = 0
                  If bNbrPressButton = 5 Then
                     Print "Vous avez presser le bouton et
la variable a une valeur de 5..."
                  ElseIf bNbrPressButton = 8 Then
                     Print "Vous avez presser le bouton et
la variable a une valeur de 8..."
                  ElseIf bNbrPressButton = 12 Then
                     Print "Vous avez presser le bouton et
la variable a une valeur de 12..."
                     bNbrPressButton = 0
                  End If
            End If

      Loop


End


'-------------------------------------------------------

Switch:

'   Incr bNbrPressButton
   bNbrPressButton = bNbrPressButton + 1
   bBoutonEvent = 1
   Toggle portb.0


Return
```

Signes ou Symboles: signification

‘	Apostrophe pour démarrer une remarque
*	Multiplication
+	Addition
-	Soustraction
,	Virgule (séparateur ex : Locate Y,X)
.	Point décimal ex : 123.45
/	Division universelle
\	Division (integer ou Word)
:	Séparateur de commande ex : **Locate** 1,2 **: LCD** " coucou "
;	Sépare les variables à afficher ex : **LCD** j ; k ; "coucou "
"	Entoure les mots à afficher ou à transférer (string)
<	Plus petit que
>	Plus grand que
=	Égal
^	Exposant : s’affiche directement avec l’accent circonflexe
>=	Plus grand ou égal
<=	Plus petit ou égal
#	Directive de compilation (voir l’aide Bascom #if #else …)
&	Précède B ou H pour spécifier la base utilisée (ex :&HC0C0) Utiliser aussi comme suffixe pour signaler des datas type long
%	Suffixe pour l’utilisation des datas intégrées
!	Utiliser pour démarrer une ligne assembleur dans le Basic
$	Directive de compilation ex : $crystal = 4000000

Pinout Arduino



Steve - VE2ONX v.1.0

Fin