Cours 01
Instructions étudiées

Configuration

- **$RegFile**
- **$Crystal**
- ***$HwStack***
- ***$SwStack***
- ***$FrameSize***

Programmation

- **Config Port**
- **Do // Loop**
- **Set // Reset**
- **Wait**
- **Dim**
- **Debounce**

$RegFile

Signale au compilateur d'utiliser le fichier spécifique des instructions correspondantes au microcontrôleur. Cette directive doit être la première instruction dans votre programme. Elle ne doit pas être écrite dans un fichier à inclure.

Exemple du fichier Dat

```
[DEVICE]
FILE=M323DEF.DAT
device = ATMEGA323
UP = M323                    ; shortname for micro
RAMSTART = $60               ; start of SRAM memory
_CHIP= 14                    ; For backwards compatibility
RAMEND  =$85F                ; Last On-Chip SRAM location
XRAMEND =$85F
E2END =$3FF
FLASHEND=$7FFF
FlashSizeText = 32 KB
SRAM = 2048                  ; SRAM size
EEPROM = 1024                ; EEPROM size
XRAMINDEX = 0                ; default no XRAM selected
XRAM = 0                     ; do not allow XRAM
WAITSTATE=0                  ; no wait state
WAITSTATEENABLE=0            ; disable setting the wait state
XRAMACCESS=0                 ; no external memory access selected
XRAMACESSENABLE=0            ; external memory access can not be selected
UBRR = 4096                  ; calculation of baudrate
```

```
[IO]
SREG        =$3f
SPH         =$3e
SPL         =$3d
GIMSK       =$3b
GICR        =$3B
GIFR        =$3a
TIMSK       =$39
TIFR        =$38
PORTA       =$1b
DDRA        =$1a
PINA        =$19
SPMCR       =$37
TWCR        =$36
MCUCR       =$35
MCUSR       =$34
MCUCSR      =$34
TCCR0       =$33
TCNT0       =$32
```

$RegFile

Un programme peut être développé sur un type de µP, puis pour une raison quelconque, évoluer vers un autre type. Avec cette directive placée en tête de programme, avant tout autre commande, on peut effectuer ces changements sans problème. Les fichiers sont tous sous la forme xxxxDEF.dat

1200DEF.DAT	attiny22.DAT	m161def.dat	m32m1.dat	usb82.dat
2313DEF.dat	attiny2313.DAT	m162def.dat	m32U2def.dat	xm128A1def.dat
2323DEF.dat	attiny2313A.DAT	m163def.dat	m32U4def.dat	xm128A3def.dat
2333DEF.DAT	attiny24.DAT	m164Pdef.dat	m406def.dat	xm128A3Udef.dat
2343DEF.dat	attiny25.DAT	m165def.dat	m48def.dat	xm128A4Udef.dat
4414DEF.dat	attiny26.dat	m168def.dat	m48Pdef.dat	xm128B1def.dat
4433DEF.DAT	attiny261.dat	m168pdef.dat	m603def.dat	xm128D3def.dat
4434DEF.dat	attiny4313.DAT	m169def.dat	m640def.dat	xm16A4def.dat
8515DEF.dat	attiny44.DAT	m16Adef.dat	m644Adef.dat	xm16D4def.dat
8535DEF.dat	attiny45.DAT	m16def.dat	m644def.dat	xm192A3def.dat
86RF401.dat	attiny461.dat	m16m1.dat	m644Pdef.dat	xm192D3def.dat
86RF401def.dat	attiny48.dat	m16U2def.dat	m645def.dat	xm256A3Bdef.dat
at12DEF.DAT	attiny84.DAT	m16U4def.dat	m649def.dat	xm256A3BUdef.dat
at15DEF.DAT	attiny85.DAT	m2560def.dat	m64def.dat	xm256A3def.dat
at22DEF.DAT	attiny861.dat	m2561def.dat	m8515.dat	xm256D3def.dat
at26def.dat	attiny88.dat	m323def.dat	m8535.dat	xm32A4def.dat
at90pwm216.dat	m103def.dat	m324PAdef.dat	m88Adef.dat	xm32A4Udef.dat
at90pwm2_3.dat	m1280def.dat	m324Pdef.dat	m88def.dat	xm32D4def.dat
attiny12.DAT	m128103.dat	m3250pdef.dat	m88pdef.dat	xm64A1def.dat
attiny13.DAT	m1281def.dat	m325def.dat	m8Adef.dat	xm64A3def.dat
attiny13A.DAT	m1284def.dat	m328def.dat	m8def.dat	xm64D3def.dat
attiny15.DAT	m1284Pdef.dat	m328pdef.dat	m8U2def.dat	xm64D4def.dat
attiny1634.dat	m128can.dat	m329def.dat	usb1287.dat	
attiny167.dat	m128def.dat	m32can.dat	usb162.dat	
attiny20.DAT	m128RFA1.dat	m32def.dat	usb646.dat	

$RegFile

Syntaxe

$REGFILE = "name"

```
'////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ____________
'
'  Ajout de $RegFile
'
'////////////////////////////////////////////////////////////////


   $regfile = "m323def.dat"                'Assigne le DEF du Micro


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

$Crystal

On peut aussi choisir la fréquence du quartz dans les options du compilateur. Il est préférable de signaler la fréquence du quartz dans le programme, ceci le rend plus visible. En revanche, il faut aussi régler la partie fuse bit. La fréquence du quartz joue sur le baud rate et sur les temps d'attente comme WAITMS.

1000000	(1 Mhz)
4000000	(4 Mhz)
6000000	(6 Mhz)
8000000	(8 Mhz)
10000000	(10 Mhz)
11059200	– (11.059200 Mhz, valeur parfait pour communication)
12000000	(12 Mhz)
14745600	– (14.745600 Mhz, valeur parfiat pour communication)
16000000	(16 Mhz, valeur courant)
20000000	(20 Mhz, quelque fois OVER CLOCK)
22118400	(22.118400 Mhz)
24000000	(24 Mhz, nouvelle version de microcontroleur)

$Crystal

Syntaxe

$CRYSTAL = var

```
'////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ____________
'
'  Ajout de $Crystal
'
'////////////////////////////////////////////////////////////////


   $crystal = 11059200                  'Assigne la frequence du crystal


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

$HwStack

Ajuste la taille de l'espace de la pile hard.

Cette fonction ne fait pas partie de ce cours. Par contre, on doit l'utiliser dans la configuration de chaque logiciel.

Valeur par défaut est : 48

Dimensionner les piles est très difficile. La plupart du temps, on y va un peu au hasard alors qu'il existe une méthode : l'utilisation de la fonction STCHECK. Voir l'aide. On peut aussi utiliser les recommandations données par les messages quand on clique sur Option/ compiler/ Chip/ framesize, HWStack...

Heureusement, les espaces Piles et frame sont réutilisables au cours du programme. Cela fonctionne un peu comme un indicateur de niveau sonore, en fonction de la demande. Aussi le plus difficile est de trouver le bon compromis.

Risques des dépassements. Le programme utilise deux fois les mêmes espaces en même temps, il y a fonctionnement aléatoire.

$HwStack

Syntax

$HwStack = var

```
'/////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  _____________
'  Ajout de $HwStack
'
'/////////////////////////////////////////////////////////////////


   $HwStack = 48


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

$SwStack

Ajuste la taille de l'espace de la pile soft.

Cette fonction ne fait pas partie de ce cours. Par contre on doit l'utiliser dans la configuration de chaque logiciel.

Valeur par défaut est : 32

Dimensionner les piles est très difficile, la plupart du temps on y va un peu au hasard, alors qu'il existe une méthode : l'utilisation de la fonction STCHECK. Voir l'aide. On peut aussi utiliser les recommandations données par les messages quand on clique sur Option/ compiler/ Chip/ framesize, HWStack...

Heureusement, les espaces Piles et frame sont réutilisables au cours du programme, cela fonctionne un peu comme un indicateur de niveau sonore, en fonction de la demande. Aussi le plus difficile est de trouver le bon compromis.

Risques des dépassements. Le programme utilise deux fois les mêmes espaces en même temps, il y a fonctionnement aléatoire.

$SwStack

Syntax

$HwStack = var

```
'////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  _____________
'  Ajout de $SwStack
'
'////////////////////////////////////////////////////////////////


   $SwStack = 32


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

$FrameSize

Ajuste la taille de l'espace cadre

Cette fonction ne fait pas partie de ce cours. Par contre on doit l'utiliser dans la configuration de chaque logiciel.

Valeur par défaut est : 32

Dimensionner les piles est très difficile, la plupart du temps on y va un peu au hasard, alors qu'il existe une méthode : l'utilisation de la fonction STCHECK. Voir l'aide. On peut aussi utiliser les recommandations données par les messages quand on clique sur Option/ compiler/ Chip/ framesize, HWStack...

Heureusement les espaces Piles et frame sont réutilisables au cours du programme, cela fonctionne un peu comme un indicateur de niveau sonore, en fonction de la demande. Aussi le plus difficile est de trouver le bon compromis.

Risques des dépassements. Le programme utilise deux fois les mêmes espaces en même temps, il y a fonctionnement aléatoire.

$FrameSize

Syntax

$FrameSize = var

```
'////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ____________
'  Ajout de $FrameSize
'
'////////////////////////////////////////////////////////////////


   $FrameSize = 32


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

Config Port

Nous allons étudier maintenant les instructions d'asservissement et de lecture des ports. Les µ-contrôleurs sont, en général, pourvus de ports multifonctions; la première étape est l'étude des ports en Entrée et Sortie. Au démarrage ou à un RESET, les ports sont mis en entrée et en haute impédance.

Les petites (de plus en plus petites) pattes des µ-contrôleurs obéissent aux ordres donnés par le programme. Il faut être très attentif à ne pas « fusiller » ces petites bêtes, par exemple, contrarier un port mis en sortie et à 1 en le court-circuitant : 5V sur 0V ! Faire attention aux ESD potentielles, donc réaliser des environnements sains, avec mise à la masse etc… Veuillez à ne pas dépasser 20mA par broche et un total de 100mA par port. Une bonne habitude, si possible, un port sortant sera équipé d'une résistance de 1KΩ, un port entrant d'une résistance de 10KΩ. 20mA sous 5volts R=U/I R=5/0.02 R=250Ω… on est tranquille avec 1KΩ.

- On doit configurer les ports (ou broche) avec CONFIG PORT, CONFIG PIN ou DDRx avant de les utiliser. (utilisation du **1°registre** des ports)
- Alors, **si et seulement si** la broche est en sortie (OUTPUT, DDRx.1), pour positionner une broche dans un état haut ou dans un état bas, on utilisera SET et RESET. Ou l'instruction PortX = n (utilisation du **2°registre** des ports).
Pour positionner l'ensemble d'un port, on préférera la fonction PORTx= Y où Y peut être un variable type binaire ou byte. Ex Portb=&B01010101, Portb=85
- Ou lire l'état des verrous d'un port : X=PortB, cette lecture ne donne pas forcément l'état actuel du port, elle donne l'image de sa dernière configuration.
- Ou lire un port en utilisant le **3°registre** des ports : Y=PINx.

Config Port

Methode 1 : Configuration de base d'un port. Configurer le port au complet.

Syntaxe

Config PortX = Input | Output

Config PinX.0 = Input / Config PinX.0 = Output

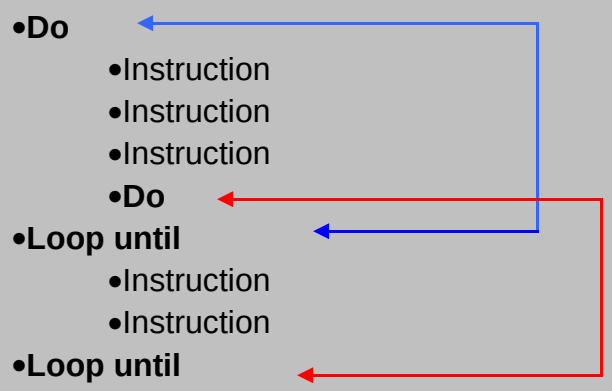
```
'////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ____________
'  Ajout de Config PortX
'
'////////////////////////////////////////////////////////////////


   Config PortA = Output

   Config PortB = Input

   Config PinB.0 = Input

   Config PinB.1 = Output

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

Do / Loop

Un programme est en général composé d'une boucle principale, et de n boucles. Nous sommes gâtés ! il y a l'embarras du choix.

Méthode simple d'utiliser une boucle.

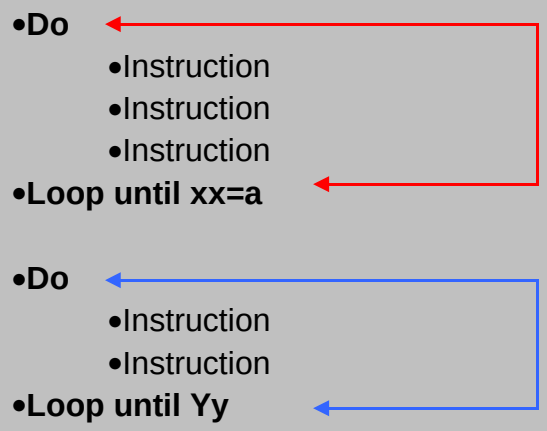
- **Remarques sur l'utilisations des boucles:**
- 1) L'imbrication de trop de boucles rend illisible le programme;
- 2) Une boucle ne peut pas se terminer hors de son champs, exemple :



- Ce programme est faux; il risque de donner des résultats…bizarres

Do / Loop

Un logiciel de microcontrôleurs a normalement une BOUCLE dans sa partie principale. C'est dans cette BOUCLE que le logiciel passera le plus de temps. C'est aussi dans cette partie que l'on executera les vérifications les plus fréquentes. Si le logiciel sort de la BOUCLE principale, il arrivera à la fin du CODE, et donc plus aucune execution à faire, c'est la FIN.



Est correct…

Do / Loop

Syntaxe

Do

...

Loop (until)

```
'////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ____________
'
'  Do / Loop
'
'////////////////////////////////////////////////////////////////

   $regfile = "m323def.dat"                'Assigne le DEF du Micro

'Fin de la configuration du Micro
'____________

     Do

       'Instruction ICI

     Loop

End
```

Set / Reset

Met un bit ou 1 Pin d'un port à 1 (Set) ou le met à 0 (reset).

Syntaxe

Reset Var.x / PinX.x

Set Var.x / PinX.x

Reset composant

```
'//////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ____________
'
'  Set  / Reset
'
'//////////////////////////////////////////////////////////////////

  $regfile = "m323def.dat"                'Assigne le DEF du Micro

'Fin de la configuration du Micro
'_____________

     Do

       Set  PortA.0
       Reset PortA.1

     Loop

End
```

Wait

Suspend le déroulement du programme pour une durée de nnn secondes.

Nnn Nombre ou une constante mais pas une variable (dans le cas WAITUS)
Les durées sont approximatives. Les interruptions ralentissent ces valeurs.

WAIT 5	Attente de 5 Secondes
WAITMS 5	Attente de 5 MilliSecondes
WAITUS 5	Attente de 5 MicroSecondes, c'est un U pas µ
Delay	Suspend le programme pendant un temps très court. La durée est d'environ 1000 µs

Wait

Syntaxe

Wait 5
WaitMs 5
WaitUs 5

```
'//////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  ---------------
'
'  Wait / WaitMs / WaitUs / Delay
'
'//////////////////////////////////////////////////////////////////

   $regfile = "m323def.dat"                  'Assigne le DEF du Micro

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

        Wait 5
        WaitMs 5
        WaitUs 5
        Delay
      Loop

End
```

Dim

Dimensionne une variable.

Utilisation classique de Dim:
Dim Jour(7) As String * 8 , tableau(25) as integer , An As Byte , Flagj As Bit Dim Lejour As String * 9 , Lemois As String * 10 , Indexjour As Byte

On peut déclarer plusieurs variables sur une seule ligne. Les 2 premières déclarations sont des tableaux de string ou de chiffre.
Le paramètre optionnel AT laisse à l'utilisateur le choix de l'emplacement mémoire où sera stocké la variable. Si l'emplacement mémoire est occupé, la variable occupera le premier emplacement libre suivant.
Overlay précise qu'une variable peut en recouvrir une autre (utiliser la même place mémoire) L'option OVERLAY n'occupe aucun emplacement mémoire, il crée un pointeur.

Dim b1 as Byte at $60 OVERLAY
Dim b2 as Byte at $61 OVERLAY
B1 et B2 ne sont pas des variables réelles! Elles pointent vers un emplacement mémoire. Dans cet exemple de &H60 et &H61, en assignant le pointeur B1, on écrira à l'adresse mémoire &H60 qui est utilisée par la variable X.

On peut aussi lire la mémoire B1: Print B1 : Cela imprimera le contenu de l'adresse mémoire &H60. En utilisant un pointeur, on peut manipuler individuellement les Bytes constituant une variable. Un autre exemple:

Dim L as Long at &H60
Dim W as Word at &H62 OVERLAY
W pointe maintenant vers les 2 Bytes supérieurs de la variable long.

Dim

Syntaxe

DIM var AS [Xram/Sram/Eram] type [AT location] [OVERLAY]

Var Un nom de variable comme varentier, j, K1 ou un d'un tableau pression(10).
Type Bit, Byte, Word, Integer, Long, Single, String.
Xram Optionnel, la variable sera stockée en mémoire externe
Sram Optionnel, la variable sera stockée en mémoire interne
Eram Optionnel, la variable sera stockée en mémoire EEPROM
AT location Optionnel, emplacement en mémoire

OVERLAY Recouvrement

Debounce

Anti-rebond sur une broche pour un switch.

Voir l'utilisation des branchements GOTO dans les "généralités“. L'instruction DEBOUNCE attend que le port change d'état. Quand cela se réalise, il y a un temps d'attente de 25 ms et un second contrôle pour éliminer les rebonds.

Si la condition est toujours bonne, il y a un branchement à l'étiquette.

Si DEBOUNCE est exécuté à nouveau, la broche doit être revenue à son état initial avant un nouveau branchement.

Chaque instruction DEBOUNCE, pour différentes broches, utilise 1 bit de mémoire pour "se souvenir" de l'état.

Debounce n’attend pas le changement d’état, pour attendre un changement d’état, utiliser DEBOUNCE avec BITWAIT

Debounce

Syntaxe
DEBOUNCE Portx.Y, etat, etiquette [, SUB]

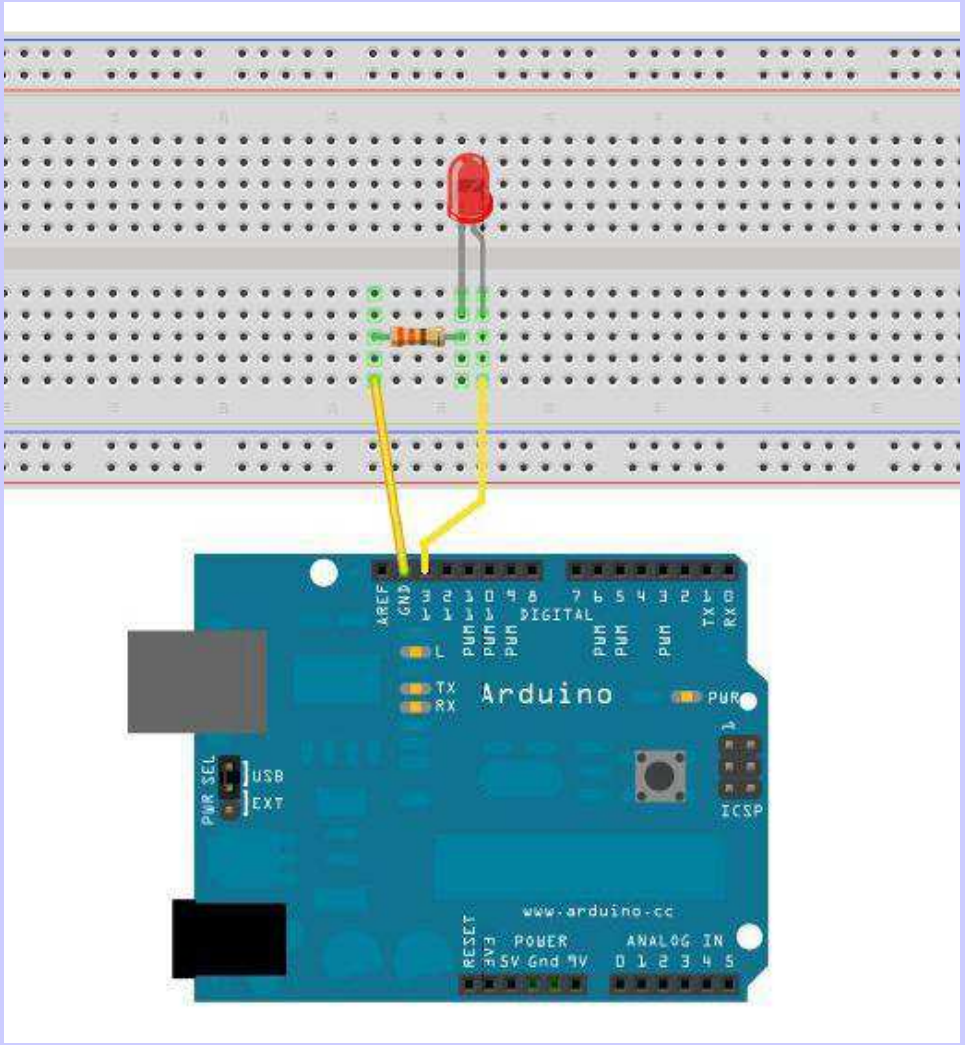
Portx.Y Une broche de port
État 0 si le branchement se fait quand la broche est à 0, ou inversement.
Étiquette: Étiquette de branchement avec un GOTO si SUB n'est pas spécifié.
SUB Branchement à l'étiquette de SUB

```
'  Ajout de Config PortX
'
'////////////////////////////////////////////////////////////////////

 Do

       Debounce pinb.5 , 0 , Switch , Sub

    Loop

End

'_____________________________

Switch:
  toggle portb.0
Return
```

Exercise #1

Faire clignoter le LED sur le circuit Arduino avec une fréquence de 1 seconde.

TIPS : Le LED est connecté sur le PortB.5 (pin 13).



Aide #1

Nouvelle commande à utiliser :

Configuration :
Ddrb
Portb.x = 0/1

Code :
Do // Loop
Toggle
Wait 1

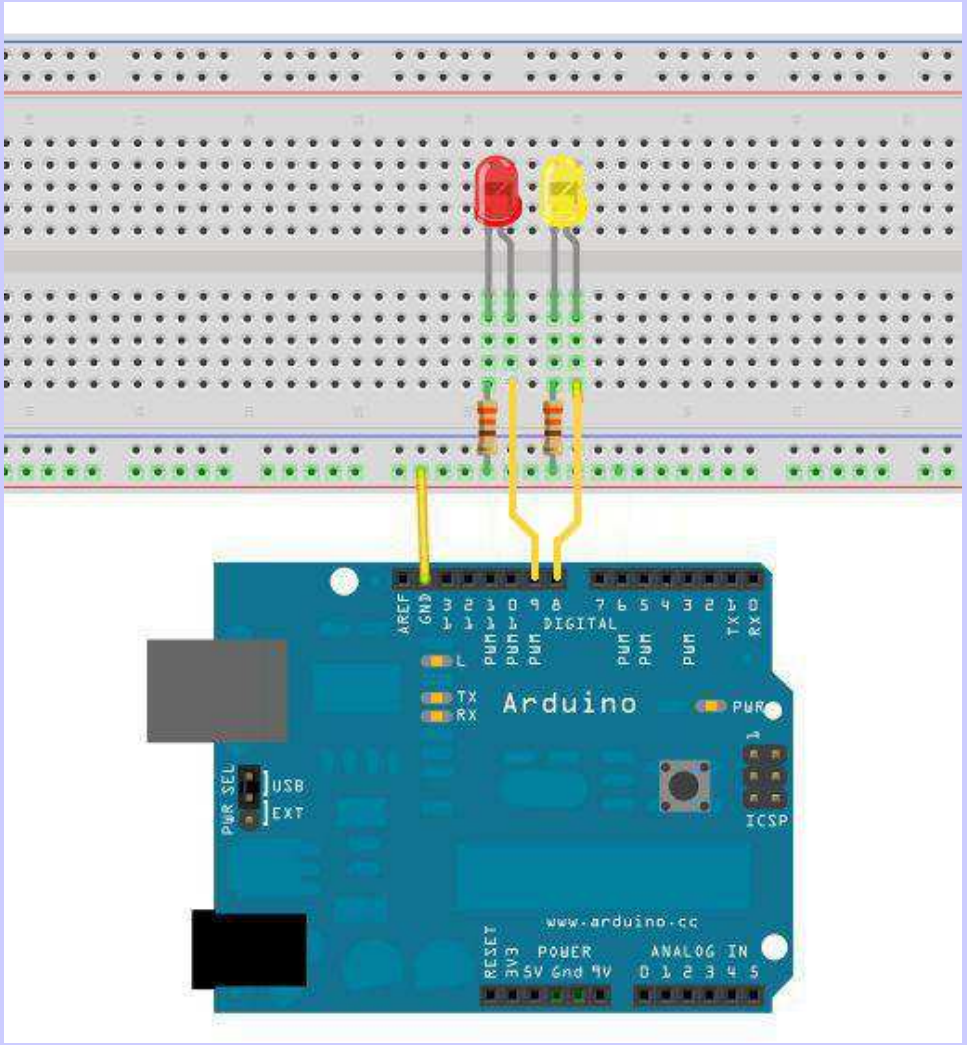
Code #1

```
'//////////////////////////////////////////////////////////////////
'
'  By Steve Tremblay
'
'  Cours # 1
'
'  Revision 1.00
'
'  FLASH 1 LED 1 seconde
'
'//////////////////////////////////////////////////////////////////

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

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

   Ddrb = &B00111111            '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


'Fin de la configuration du Micro
'_____________


    Do
      Toggle portb.5
      Wait 1
    Loop

End
```

Exercise #2

Faire clignoter 2 LEDS avec une alternance de 1 seconde et décaler de ½ seconde

TIPS : Diviser le temps en deux...



Aide #2

Nouvelle commande à utiliser :

Configuration :

Code :
Set // Reset
WaitMs

Code #2

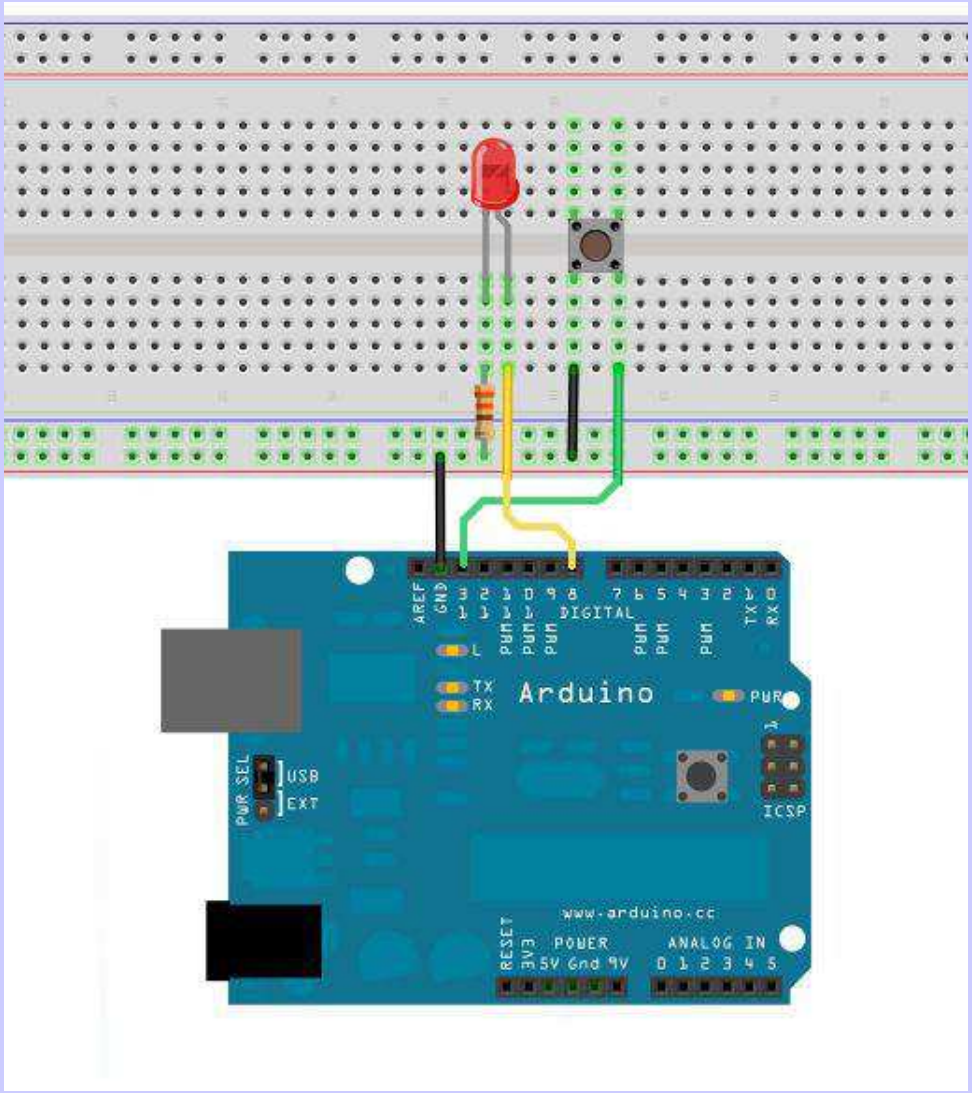
```
'////////////////////////////////////////////////////////////////
'
' By Steve Tremblay
'
' Cours # 1
'
' Revision 1.00
'
' ____________
'
' FLASH 2 LEDS 50% delais
'
'////////////////////////////////////////////////////////////////


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

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

  Ddrb = &B00111111          '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

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

```
Do

    Set portb.0
    Waitms 500

    Set portb.1
    Waitms 500

    Reset  portb.0
    Waitms 500

    Reset portb.1
    Waitms 500

  Loop


End
```

Exercise #3

Faire changer l'état d'un LED avec un interrupteur.

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



Aide #3

Nouvelle commande à utiliser :

Configuration :
CONFIG DEBOUNCE = temps (en milliseconde) par default 25 Ms

Code :
Debounce PinX.x
(Label)

Code #3

```
'////////////////////////////////////////////////////////////////////
'
' By Steve Tremblay
'
' Cours # 1
'
' Revision 1.00
'
' ____________
'
' FLASH 1 LED avec 1 SW
'
'////////////////////////////////////////////////////////////////////

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

   $HwStack = 48
   $SwStack = 32
   $FrameSize = 32
   $Crystal = 16000000
   Config  DEBOUNCE = 25

   Ddrb = &B00111111            '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


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

```
   Do

      Debounce pinb.5 , 0 , Switch , Sub

   Loop

End


'_____________________________________

Switch:
  Toggle portb.0
Return
```

Fin