```
//arlotto 2007 : routine liaison série avec it en rx pour pic 18F
// Ce fichier montre l'émission et la réception de chaîne de caractères.
// L'émission sur fait simplement par printf ou WriteUSART ou putsUART
// La réception d'une chaîne se terminant par CR (0x0D) est ici faite dans
// une routine d'IT
// ici les it sont utilisées dans le mode compatible mid range (pic16)
// pas de priorité (RCONbits.IPEN = 0 )
// L'autorisation se fait par les bits GIE et PEIE de INTCON.
// Attention à la gestion particulière des chaînes constantes par PIC18
// (cf C18 user guide DS51288e chap 2.7.3)
// le message "ON" allumme RB1 le message OFF l'éteind
// le message "CLI nnnn" fait clignoter RB2 un tour de boucle sur nnnn fois
// 1000<=nnnn<=20000
// l'appui sur RB0 affiche un X

// configurer le terminal 9600 N 8 1 pas de contrôle de flux transmission de CR
en fin de ligne

// Compiler le programme avec l'option Memory Model "Large code model".

#include <p18cxxx.h>
#include <usart.h>   // pour fonctions UART
#include <string.h>  // pour strmcp
#include <stdio.h>  // pour printf
#include <stdlib.h> // pour atoi
// configuration PICDEM2+ quartz
#pragma config OSC = HS
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config PBADEN = OFF  // Si PBADEN = ON RB4:RB0 sont analogiques au reset
!

#define Button PORTBbits.RB0
#define GreenLed PORTBbits.RB1
#define TrisGreenLed TRISBbits.TRISB1
#define RedLed   PORTBbits.RB2
#define TrisRedLed   TRISBbits.TRISB2

#define MAX_MSG 20
#define CAR_CR 0x0D
#define CAR_LF 0x0A
#define CAR_BS 0x08

#define CLI_DEF 8000

//prototype du gestionnaire d'interruption
void InterruptHandlerHigh (void);

volatile char r = 0 ; // indique qu'une chaîne complète à été reçue
volatile char msg[MAX_MSG+1]; // contient la chaîne reçue


void main(void){
                    int x=0;
                    char c;
                    int cli = CLI_DEF ;
                    int tmp_cli ;
                    char i=0;
                    char state=0;
                    GreenLed= 0 ;
                    TrisGreenLed = 0 ;
                    RedLed = 0 ;
                    TrisRedLed  = 0 ;
```

```
// init 9600Bd à 4Mhz avec it en rx
OpenUSART( USART_TX_INT_OFF &
USART_RX_INT_ON &
USART_ASYNCH_MODE &
USART_EIGHT_BIT &
USART_CONT_RX &
USART_BRGH_HIGH,
25 );
INTCONbits.PEIE = 1 ; // autorisation des it périphériques
INTCONbits.GIE = 1 ; // autorisation globale des it

printf("Demo USART PIC18\r\n");
printf("Commandes : ON / OFF / CLI <p> (1000<=p<=20000)\r\n>");

for( ;; )
{

  //émission sur bouton :
 switch (state)
 {
  case 0 :
       if(!Button) { state=1;}
       break ;
  case 1 :
   while(BusyUSART());
   WriteUSART('X');
   state=2 ;
   break ;
  case 2 :
      if(Button) { state=0;}
       break ;

 }

/* Interprèteur de commandes minimal */
// exécution de la commmande si chaîne compléte reçue
// attention à bien utiliser la bonne version de strcmp
// en fonction de la nature (ram ou rom) des paramètres

 if(r){
     if(strcmppgm2ram(msg,"ON")==0 ) {
                        GreenLed = 1 ; }
     else {
         if(strcmppgm2ram(msg,"OFF")==0 ) {
                        GreenLed = 0 ; }
    else {
      c=msg[3];
      msg[3]='\0'; // place fin de chaîne après CLI
      if(strcmppgm2ram(msg,"CLI")==0 ) {
      tmp_cli = atoi(msg+4); // conv param en int
      if(tmp_cli >=1000 && tmp_cli<=20000)
      {cli=tmp_cli;}
       else { printf("CLI : Valeur %d interdite",tmp_cli);}
      }
     else {
          msg[3]=c; // restitue la chaîne
          printf("Unknown command :%s \x7",msg);
          }
          }
         }
     printf("\r\n>");
     r=0;
    }
```

```
 // clignotement le de vie (variable en fonction de cli)
x++;
if (x>=cli){
RedLed = !RedLed ;
x=0; }
}
}

//Placement d'un saut vers le gestionnaire d'interruption
//-----------------------------------------------------------------------------
// High priority interrupt vector
#pragma code InterruptVectorHigh = 0x08
void
InterruptVectorHigh (void)
{
  _asm
    goto InterruptHandlerHigh //jump to interrupt routine
  _endasm
}

// Routine gestionnaire d'interruption
//-----------------------------------------------------------------------------
// High priority interrupt routine
#pragma code
#pragma interrupt InterruptHandlerHigh
void
InterruptHandlerHigh ()
{
   static char i ; // doit être statique pour conserver sa valeur entre les IT
   char c ;
  // Partie réception d'un caractère
  if(PIR1bits.RCIF) // si un car arrivé
    {
      c=ReadUSART(); // le lire (fait repasser RCIF à zéro)
      if(RCSTAbits.FERR || RCSTAbits.OERR) {
             RCSTAbits.CREN = 0 ; RCSTAbits.CREN= 1 ; }
      while(BusyUSART()); // par sécurité
      WriteUSART(c); // echo
      if( c==CAR_BS)
        {
         if(i>0){
             i--;  }
        }
      else {
        if(c!=CAR_CR && i<MAX_MSG)
         {
           msg[i++]=c ; // stockage
         }else
         {
           msg[i]='\0'; // fin de chaîne si CR
           i=0;
           r=1;
         }
      }
    }
 // Placer ici les autres parties
 // if(Autre bit F)
  // {
  //   Raz du bit F ;
  //   Traitement ;
  // }
}
```