```
// RecepteurTVsport.c
// Robert Denoyel
// Le PIC est en horloge interne sans Clock Out( 4 MHz)          #FUSES INTRC_IO   //Internal RC Osc, no CLKOUT
// La patte MCLR est disponible pour le reset                    #FUSES MCLR       //Master Clear pin enabled
// Le multiplexage de l'affichage est géré par timer0 (extinction et allumage pendant 8,2ms)
// Le coefficient multiplicateur se fait par positionnement de SW1 (SW1_H, SW1_L)  de COEF MULT ( I1 )
// 0ff, Off--> test(-->*0)  Off, On  -->*1  On, Off-->*2  On,On-->*4
// Le multiplexage de l'affichage est géré par timer0 (extinction et allumage pendant 8,2ms)
// Timer2 permet le comptage des secondes ( incrementation de temps_TV)
// temps_restant de TV est affichee sur AFFG et AFFD
// La recepteur recoit des donnees sur RX ( PB1)
// La commande du relais se fait par PB0

/* Inclusions et équivalences */
#include "ReceptSportTV 16F628.h"
#define selaff_gauche pin_B5                  // Sélection de l'afficheur gauche
#define selaff_droit pin_B4                   // Sélection de l'afficheur droit
#define SW1_L input(pin_A7)                        // Entree de coef multiplicateur (LSB)
#define SW1_H input(pin_B3)                        // Entree de coef multiplicateur (MSB)
#define RXPIC input(pin_B1)                   // entree de reception de PIC Emetteur
#define Commande_relais pin_B0                // sortie de commande du relais

/* Déclaration des variables globales */
int cathodes_gauche=0b00000000;             // Mot de commandes des cathodes de l'afficheur gauche initialisé à éteint (1 --> allumé)
int cathodes_droit=0b00000000;              // Mot de commandes des cathodes de l'afficheur droit initialisé à éteint (1 --> allumé)
int choix_affich=0;                         // Variable de choix d'afficheur (0 -->gauche )
int temp=0;                                 // variable temporaire
int32 temps_sport=15;                       // variable de calcul du temps de sport en secondes (9h50mn max-->35400s max)
int32 temps_TV=0;                           // variable de calcul du temps d'usage TV en secondes
int32 temps_restant=0;                      // variable de calcul du temps restant de TV en secondes
int coef=1;                                 // coefficient de calcul de temps de TV
int Nbre_fois=0;                            // variable de calcul du nombre d'interruption de timer2 (100-->1s)
int heures=0;                               //  nombres d' heures de sport
int minutes=0;                              // nombres de minutes de sport
int secondes=0;                             // nombres de secondes de sport
int AffG=0;                                 // chiffre gauche du temps compte ( commandes 7seg pour le 0,1....9 )
int AffD=0;                                 // chiffre droit du temps compte  ( commandes 7seg pour le 0,1....9 )
int G=0;                                    //chiffre gauche du temps compte à afficher en binaire
int D=0;                                    //chiffre droit du temps compte à afficher en binaire
int
tab[10]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,0b01101101,0b01111100,0b00000111,0b01111111,0b01100111};
//table de conversion
char trame_rx[5]={0,0,0,0,0};               //tableau  de reception de trame
int nboctet=0;                              // variable d'envoi de trame
int32 timeout=0;                            // variable de debordement d'attente de reception
int1 timeout_error=0;                       // variable d'erreur de reception

/* Déclaration des prototypes */
void affiche(void);                         //pas de variable d'entrée et de  sortie
void convertit_temps(int32 val);            //pas de variable d'entrée et de  sortie
int lit_char(void);                         //pas de variable d'entrée et 1 variable de sortie
void recoit_emetteur(void);                 //pas de variable d'entrée et de  sortie

/* Déclaration des fonctions */

/* Fonction affiche: affiche le contenu de cathodes_gauche ou cathodes_droit
sur l'afficheur gauche ou droit en fonction de choix_affich */
void affiche(void)
{
  if (choix_affich==1)
     {
       temp=cathodes_gauche;
       temp=temp & 0b00000001;
       if(temp!=0)output_low(Pin_A0); else output_high(Pin_A0);
       temp=cathodes_gauche;
       temp=temp & 0b00000010;
       if(temp!=0)output_low(Pin_A1); else output_high(Pin_A1);
       temp=cathodes_gauche;
       temp=temp & 0b00000100;
       if(temp!=0)output_low(Pin_A2); else output_high(Pin_A2);
       temp=cathodes_gauche;
       temp=temp & 0b00001000;
       if(temp!=0)output_low(Pin_A3); else output_high(Pin_A3);
       temp=cathodes_gauche;
       temp=temp & 0b00010000;
       if(temp!=0)output_low(Pin_A4); else output_high(Pin_A4);
       temp=cathodes_gauche;
       temp=temp & 0b00100000;
```

```
      if(temp!=0)output_low(Pin_A6); else output_high(Pin_A6);
      temp=cathodes_gauche;
      temp=temp & 0b01000000;
      if(temp!=0)output_low(Pin_B6); else output_high(Pin_B6);
      temp=cathodes_gauche;
      temp=temp & 0b10000000;
      if(temp!=0)output_low(Pin_B7); else output_high(Pin_B7);
      output_low(selaff_gauche);              // validation de l'afficheur gauche
      output_high(selaff_droit);              // Inhibition de l'afficheur droit
    }
  else
    {
      temp=cathodes_droit;
      temp=temp & 0b00000001;
      if(temp!=0)output_low(Pin_A0); else output_high(Pin_A0);
      temp=cathodes_droit;
      temp=temp & 0b00000010;
      if(temp!=0)output_low(Pin_A1); else output_high(Pin_A1);
      temp=cathodes_droit;
      temp=temp & 0b00000100;
      if(temp!=0)output_low(Pin_A2); else output_high(Pin_A2);
      temp=cathodes_droit;
      temp=temp & 0b00001000;
      if(temp!=0)output_low(Pin_A3); else output_high(Pin_A3);
      temp=cathodes_droit;
      temp=temp & 0b00010000;
      if(temp!=0)output_low(Pin_A4); else output_high(Pin_A4);
      temp=cathodes_droit;
      temp=temp & 0b00100000;
      if(temp!=0)output_low(Pin_A6); else output_high(Pin_A6);
      temp=cathodes_droit;
      temp=temp & 0b01000000;
      if(temp!=0)output_low(Pin_B6); else output_high(Pin_B6);
      temp=cathodes_droit;
      temp=temp & 0b10000000;
      if(temp!=0)output_low(Pin_B7); else output_high(Pin_B7);
      output_high(selaff_gauche);             // Inhibition  de l'afficheur gauche
      output_low(selaff_droit);               //  validation de l'afficheur droit
    }
}

// Fonction convertit_temps: convertit le temps ( valeur en secondes) en heures, minutes, secondes
// Gere l'affichage en unité d'heure et dizaines de minutes ou en dizaines et unites de minutes ou en dizaines et unites de secondes ( G, D )
// Gere les commandes des leds correspondantes (AFFG, AFFD )
void convertit_temps(int32 valeur)
{
  heures=valeur/3600;
  minutes=valeur/60-heures*60;
  secondes=(valeur-heures*3600-minutes*60);
  If ((heures==0) && (minutes==0))          // il reste des secondes de temps--> --> 2 points
    {
      G=secondes/10;
      D=secondes-G*10;
      AffD=tab[D]+128;
      AffG=tab[G]+128;
      output_high (Commande_relais);
    }
  else
    { If ((heures==0) && (minutes!=0))        // il reste des minutes de temps--> 1 point
        {
          G=minutes/10;
          D=minutes-G*10;
          AffD=tab[D]+128;
          AffG=tab[G];
          output_high (Commande_relais);
        }
      else
        {
        If (heures<10)                  // il reste des heures de temps
          {
            G=heures;
            D=minutes/10;
            AffD=tab[D];
            AffG=tab[G];
            output_high (Commande_relais);
          }
        else                            // temps restant=0 donc affichage 8.8.temps  max = 9H50
          {
```

```
            AffG=0b11111111;
            AffD=0b11111111;
            output_low (Commande_relais);
            }
          }
      }
}

/* déclaration du programme d'interruption de débordement du timer0 (muxliplexage de l'affichage=8,2ms) */
#int_RTCC
RTCC_isr()
{
  disable_interrupts(INT_RTCC);                    // inhibition des interruptions du timer1
    if (choix_affich==0)
      choix_affich=1;
    else
      choix_affich=0;
  affiche();
  enable_interrupts(INT_RTCC);                     // validation des interruptions du timer1
}

// déclaration du programme d'interruption de débordement du timer2 (interruption toutes les 10ms)
//  On incrémente temps_TV toutes les secondes     On calcule temps_restant
#int_TIMER2
TIMER2_isr()
{
  disable_interrupts(INT_TIMER2);        // inhibition des interruptions du timer2
  if (Nbre_fois<=100)                    // temps écoulé<=1s?
    Nbre_fois=Nbre_fois+1;
  else                                   // temps écoulé=1s!!!!!!
    {
      temps_TV=temps_TV+1;
/*        switch (coef)
        {
          case 0: temps_restant=temps_sport*1-temps_TV;
            break;
          case 1: temps_restant=temps_sport*2-temps_TV;
            break;
          default: ;
            break;
        }
*/
      temps_restant=temps_sport*coef-temps_TV;
      convertit_temps(temps_restant);
      cathodes_gauche=AffG;              // affichage du temps compté sur AFFG
      cathodes_droit=AffD;               //  AFFD eteint
      Nbre_fois=0;
    }
  enable_interrupts(INT_TIMER2);                   // validation des interruptions du timer2
}

// Réception d'un caractère sur liaison série ou sortie sur timeout
int lit_char()
{
  timeout = 0;
  timeout_error = 0;

  while (!kbhit() && (++timeout < 15000))      // Attente tant que pas caractère
    delay_ms(1);                           // reçu et temps < 15 s

  if (kbhit())                             // Si caractère reçu
    return (getc());                       // retour caractère lu
  else
    {
    timeout_error = 1;                     // Sinon Erreur timeout
    }
}

// Réception trame de la carte Emetteur
void recoit_emetteur()
{
    nboctet = 0;
    timeout_error = 0;
    while ((timeout_error == 0) && (nboctet < 5))     // on attend 5 octets
      {
        trame_rx[nboctet] = lit_char();    // Réception du caractère
        nboctet++;
      }
```

```
    if (timeout_error == 0)              // si trame reçue (pas timeout)
      {
        nboctet = 0;
        // if (trame_rx[0] == 'A')          // si trame OK     // temps_sport <---reception
        temps_sport=trame_rx[4]+trame_rx[3]*256+trame_rx[2]*62536+trame_rx[1]*16777216+20;  //+20 pour combler le retard de 1ere
émission
      }
    else
      {
        cathodes_gauche=0b01111001;          // pas de reception durant 15s: Affiche Er
        cathodes_droit=0b01010000;
        delay_ms(1000);
      }
}

/* Programme Principal */
void main()
{
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_32);         // timer0 interruption au bout de 8,2ms
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DIV_BY_4,156,16);               // Timer2 regle à 10ms
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);
  disable_interrupts(INT_TIMER1);                  // Pas d'interruption de timer1
  disable_interrupts(INT_RDA);                     // Pas d'interruption à la reception de donnees
  disable_interrupts(INT_EXT);                     // Pas d'interruption externe
  enable_interrupts(INT_RTCC);                     // validation des interruptions du timer0 ( 8,2ms)
  enable_interrupts(INT_TIMER2);                   // validation des interruptions du timer2 ( 10ms )
  enable_interrupts(GLOBAL);
  SET_TRIS_B(0b00001010);                          //E/S de PB
  SET_TRIS_A(0b10100000);                          //E/S de PA
  do
    {
      if ((SW1_H==1)&&(SW1_L==0))      // Mode1==> Off-On  ( 01 )
        coef=1;
      else;
      if ((SW1_H==0)&&(SW1_L==1))      // Mode2==> On-Off  ( 10 )
        coef=2;
      else;
      if ((SW1_H==0)&&(SW1_L==0))      // Mode3==> On-On   ( 11 )
        coef=4;
      else;
      if ((SW1_H==1)&&(SW1_L==1))      // Mode0==> Off-Off ( 00 )
      coef=0;
      recoit_emetteur();
      delay_ms(50);
 }
    While( true);
}
```