```
/*Programme VFO_FT-707 par F1HKJ. Dernière modification le 15/08/2015

 Prévu pour fonctionner avec un Arduino uno
 Générateur HF avec DDS AD9851 pour utilisation en VFO du FT-707
 Fréquence de 2 à 42 MHz (12 à 40 MHz avec carte et coffret VFO FT-707)
 Gestion fréquence par touches et encodeur rotatif
 Pas de 1 Hz à 1000 KHz (pas min modifiable par soft)
 Gestion USB/LSB/CW-AM modifiable par soft
 Sauvegarde de la configuration à l'arrêt du système
 Calibration de l'horloge par soft
*/
  #include <LiquidCrystal.h>   //Librairie LCD
  #include <EEPROM.h>          //Librairie eeprom

//----------Constantes à modifier selon le matériel et ses préférences----------

  #define DDS_REF 120000750    // Fréquence exacte de l'horloge 120 MHz du DDS (Hz)
  #define F_max 40000000        // Fréquence maximum du DDS (Hz)
  #define F_min 12000000        // Fréquence minimum du DDS (Hz)
  #define F_FI   8987500        // Fréquence centrale filtre FI (Hz) (pour mémoire)
  #define F_lsb  8986000        // Fréquence porteuse FI mode LSB (Hz)
  #define F_usb  8989000        // Fréquence porteuse FI mode USB (Hz)
  #define F_cw_am 8988300       // Fréquence porteuse FI mode CW-AM (Hz)
  #define T_RC       10         // tempo répétition roue codeuse (ms)
  #define T_touche   200        // tempo répétition touches (ms)
  #define P_min      1          // Pas minimum en fréquence (Hz)
  #define seuil    1018         // seuil alim en dessous duquel on provoque une sauvegarde

//--------------------------------------------------------------------------------
// N-B : Les broches Arduino 14 à 19 du logiciel correspondent
// aux broches AO à A5 du schéma et de la carte Arduino uno.
//--------------------------------------------------------------------------------

// Affectation des Pins Arduino pour Load, Clock et Data du DDS AD9851

  #define DDS_LOAD  8          // FQ_UD (validation)
  #define DDS_CLOCK 7          // W_CLK (horloge)
  #define DDS_DATA  6          // DATA  (données)

// Affectation des Pins Arduino pour afficheur LCD, poussoirs et encodeur

  #define PinA  2              // entrée encodeur PinA (interruption)
  #define PinB  17             // entrée encodeur PinB (sens)

  LiquidCrystal lcd(12, 11, 5, 4, 3, 16);  // affectation pins afficheur
```

```
  int usblsb = 10;              // switch usb-lsb en pin 10
  int toucheup = 14;            // touche up en pin 14
  int touchedown = 15;          // touche down en pin 15
  int touchepas = 9;            // touche pas en pin 9
  int cwblu = 13;               // touche blu/cw-am en pin 13

//----------------------Variables et Flags divers----------------------------

  unsigned long freq;           // Fréquence du DDS
  unsigned long frequaffich;    // pour affichage LCD, Fréquence FT-707
  unsigned long frequaf;        // idem
  long Pas;                     // valeur du pas à ajouter ou retrancher
  int alim = A4;                // Entrée en A4 de la mesure de la tension d'alim 8V
                                // (pour sauvegarde en eeprom si au dessous du seuil)



  unsigned char un,deux,trois,quatre;   // les quatre poids du long "freq"
  unsigned char TST_eeprom;             // lecture test à l'adresse 5 de l'eeprom
                                        // ( = "FFh" si premier démarrage)

  volatile boolean mouvement;       // signale un mouvement de la roue codeuse
  volatile boolean UpDown;          // signale le sens de la roue codeuse
  volatile boolean LSB;             // mode USB/LSB
  volatile boolean action;          // signale si une touche a été enfoncée
  volatile boolean cw_blu;          // mode blu/cw-am

//-------Routine d'interruption 0 déclenchée quand PinA passe de high à low--------

void Interruption_RC()
{
  mouvement = true;                 // il y a eu mouvement
  UpDown = (digitalRead(PinB));     // lecture du sens
}

//-----Routine de sauvegarde des données Pas et freq en eeprom à la coupure alim----

void Sauvegarde()
 {
  //----------Sauvegarde du Pas--------------
  char cptr = 0;
  while(Pas >0)
  {
   cptr++;                          // cptr = 1 + puissance de 10 du pas
   Pas = Pas/10;
  }
  EEPROM.write(5, cptr);        // sauvegarde en eeprom à l'adresse 5
```

```
  //----------Sauvegarde fréquence-----------
  un = freq & 0x000000FF;                  // isolement du poids un (poids faible)
  deux = (freq >> 8) & 0x000000FF;         // isolement du poids deux
  trois = (freq >> 16) & 0x000000FF;       // isolement du poids trois
  quatre = (freq >> 24) & 0x000000FF;  // isolement du poids quatre (poids fort)

  EEPROM.write(1, quatre);              // écriture poids 4, adresse 1
  EEPROM.write(2, trois);               // écriture poids 3, adresse 2
  EEPROM.write(3, deux);                // écriture poids 2, adresse 3
  EEPROM.write(4, un);                  // écriture poids 1, adresse 4


  while((analogRead(alim)) <= seuil)    // on boucle tant que Ualim < seuil
  {
  }
 }                                              // sinon on retourne dans loop

//----------Routine lecture au démarrage des données Pas et freq en eeprom----------

void Lecteeprom()
{
  //--------------Lecture du Pas au démarrage--------------
  char cptr;
  cptr = EEPROM.read(5);               // valeur du pas à l'adresse 5
  Pas = 1;                             // pas = 10 puiss(0) (cptr=1)
  while(cptr > 1)
  {
    cptr--;
    Pas = Pas*10;                      // calcul du pas = 10 puiss(cptr-1)
  }
  //-------------Lecture de la fréquence au démarrage------
  unsigned char fort = EEPROM.read(1);     // valeur du poids quatre (fort) à l'adresse 1
  unsigned char forta = EEPROM.read(2);    // valeur du poids trois à l'adresse 2
  unsigned char fortb = EEPROM.read(3);    // valeur du poids deux à l'adresse 3
  unsigned char faible = EEPROM.read(4);   // valeur du poids un (faible) à l'adresse 4

  //calcul de la fréquence au démarrage en utilisant les 4 poids en puissances de 256
  freq = fort;                         // fort x 256 puiss 3
  freq = (freq << 8) + forta;          // +forta x 256 puiss 2
  freq = (freq << 8) + fortb;          // +fortb x 256 puiss 1
  freq = (freq << 8) + faible;         // +faible x 256 puiss 0 (x1)
}

//------------Routine d'initialisation au démarrage----------------------

void setup()
```

```
{
  pinMode(PinA, INPUT);                 // Entrée interruption roue codeuse
  pinMode(PinB, INPUT);                 // Sens roue codeuse
  pinMode(toucheup, INPUT);             // poussoir "UP"
  pinMode(touchedown, INPUT);           // poussoir "DOWN"
  pinMode(touchepas, INPUT);            // poussoir "PAS"
  pinMode (usblsb, INPUT);              // inverseur USB/LSB
  pinMode (DDS_DATA,  OUTPUT);          // DATA (Données DDS)
  pinMode (DDS_CLOCK, OUTPUT);          // W_CLK (Horloge DDS)
  pinMode (DDS_LOAD,  OUTPUT);          // FQ_UD (validation DDS)
  pinMode (cwblu, INPUT);               // inverseur blu/cw-am


//---------------- Affichage lors de l'initialisation  --------------------
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print ("VFO DDS 12-40MHz ");
  lcd.setCursor(0, 1);
  lcd.print( "     F1HKJ");

  delay(4000);                          // tempo affichage message (4s)

  lcd.setCursor(0, 0);
  lcd.print ("              ");
  lcd.setCursor(0, 1);
  lcd.print ("              ");

//---------------Entrée valeurs fréquence et pas de départ----------------

 TST_eeprom = EEPROM.read(5);
 if (TST_eeprom == 255)     // si c'est le premier démarrage, valeur en eeprom = FFh
 {
  Pas = 1000;                // alors pas = valeur par défaut
  freq = 18989000;           // et fréquence = valeur par défaut
 }
 else                        // sinon on exécute la routine de lecture eeprom
 {
  Lecteeprom();              // lecture pas de départ et fréquence au démarrage
 }

 Affich_pas();
 LSB = false;                // mode temporaire au démarrage (sera modifié au
                             // premier tour de boucle si inverseur sur LSB)

 Affich_freq();             // appel routine calcul et affichage de la fréquence
 frequency(freq );          // appel routine calcul et contrôle AD9851
```

```
//-----------------init flags et variables diverses-----------------------

  mouvement = false;      // Pas de mouvement au démarrage
  action = false;         // aucune action en cours

//---------init interruption "0" (pin2) sur front descendant PinA----------

  attachInterrupt(0,Interruption_RC,FALLING);
}

// FIN de l'initialisation

//-----------boucle principale (scrutation touches et actions)-------------

void loop()
{

  if (analogRead(alim) <= seuil)        // si alim 8V < seuil
  {
    Sauvegarde();                       // sauvegarde des données Pas et fréquence
  }
 // Ne continue ici que si la tension d'alim est supérieure au seuil
 // ou si elle est remontée pendant la sauvegarde

  if (mouvement)                 // si roue codeuse a été actionnée
  {
    if (UpDown)                  // et si pinB était à "un"
    {
      if (freq > F_min)          // et si pas fréquence min
      {
       freq = (freq - Pas);      // alors descente
       action = true;            // pour action
      }
    }
    else                         // si pinB était à "zéro"
    {
      if (freq < F_max)          // et si pas fréquence max
      {
        freq = (freq + Pas);     // alors montée
        action = true;           // pour action
      }
    }
  }

  //----------------- Lecture touches --------------------

  int appui_pas = digitalRead(touchepas);
```

```
int appui_up = digitalRead(toucheup);
int appui_down = digitalRead(touchedown);
int toggle_LSB = digitalRead(usblsb);
int cwoublu = digitalRead(cwblu);

//---------------- traitement touches --------------------

if(appui_pas == LOW)          // si appui sur touche "pas"
{
  action = true;              // pour action
  Pas = (Pas/10);             // uniquement sens décroissant
  if (Pas < P_min)            // si < pas minimum
    Pas = 1000000;            // alors pas maximum
  Affich_pas();               // appel routine d'affichage
}
if(appui_up == LOW)           // si appui sur touche "up"
{
  if (freq < F_max)           // et si pas fréquence max
  {
    freq = (freq + Pas);      // alors montée
    action = true;            // pour action
  }
}
if(appui_down == LOW)         // si appui sur touche "down"
{
   if (freq > F_min)          // et si pas fréquence min
   {
     freq = (freq - Pas);     // alors descente
     action = true;           // pour action
   }
}

if (cwoublu == !cw_blu)       // si inverseur blu/cw-am modifié
{
  cw_blu = !cw_blu;           // alors on change le mode
  action = true;              // pour action
}

if (toggle_LSB == !LSB)       // si inverseur USB/LSB modifié
{
   LSB = !LSB;                // alors on change le mode
   action = true;             // pour action
}

if (action)                   // si action a été demandée
{
  action = false;             // raz flag action
```

```
    Affich_freq();            // Appel routine Affichage mode et fréq. FT-707
    frequency(freq);          // Appel routine calcul delta phi et envoi au DDS
    if (mouvement)            // si la roue codeuse a été actionnée
    {
      delay(T_RC);            // délai répétition et anti-rebonds roue codeuse
      mouvement = false;      // reset roue codeuse
    }
    else                      // si ce n'était pas la roue codeuse
      delay(T_touche);        // délai plus long pour les touches
  }
}
// FIN de la boucle principale (pas de tempo si tous les tests ont été négatifs)

//--------------Calcul du delta-phi (vs frequency) et envoi au DDS AD9851-----------------

void frequency(unsigned long frequency)
{
  unsigned long tuning_word = (frequency * pow(2, 32)) / DDS_REF;  // calcul delta-phi DDS

  digitalWrite (DDS_LOAD, LOW);     // init pin Load niveau bas

  int i;

  for(i = 0; i < 32; i++)           // pour les 32 bits
  {
    if ((tuning_word & 1) == 1)     // si le bit est à 1
      outOne();
    else                            // sinon il est à zéro
      outZero();
    tuning_word = tuning_word >> 1;     // bit suivant
  }
  byte_out(0x08);                   // + 8 bits de service
  digitalWrite (DDS_LOAD, HIGH);        // remet la Pin "Load" au niveau haut (validation)
}

//--------------------------envoi d'un octet au DDS-----------------------------------

void byte_out(unsigned char byte)
{
  int i;
  for (i = 0; i < 8; i++)           // pour les 8 bits
  {
    if ((byte & 1) == 1)            // si le bit est à 1
      outOne();
    else                            // sinon il est à zéro
      outZero();
    byte = byte >> 1;               // bit suivant
```

```
  }
}

//-------------------envoi d'un bit DATA à "1" au DDS avec horloge----------------------

void outOne()                           // envoi d'un bit "1" au DDS
{
  digitalWrite(DDS_CLOCK, LOW);
  digitalWrite(DDS_DATA, HIGH);
  digitalWrite(DDS_CLOCK, HIGH);
  digitalWrite(DDS_DATA, LOW);
}

//--------------------envoi d'un bit DATA à "0" au DDS avec horloge----------------------

void outZero()                           // envoi d'un bit "0" au DDS
{
  digitalWrite(DDS_CLOCK, LOW);
  digitalWrite(DDS_DATA, LOW);
  digitalWrite(DDS_CLOCK, HIGH);
}

//------------------------------Affichage du pas-------------------------------------

void Affich_pas ()
{

  if (Pas < 1000)                  // 1, 10, 100
  {
    lcd.setCursor(0, 1);
    lcd.print("              ");
    lcd.setCursor(5, 1);
    lcd.print ("Pas ");
    lcd.print (Pas);
    lcd.print (" Hz");
  }
  else                             // 1000 et au dessus
  {
    lcd.setCursor(0, 1);
    lcd.print("              ");
    lcd.setCursor(5, 1);
    lcd.print ("Pas ");
    lcd.print (Pas/1000);
    lcd.print ("KHz");
  }
}
//-----------------------Affichage Fréquence HF FT-707 et mode-------------------------
```

```
void Affich_freq ()
{

// Calcul et affichage de la fréquence porteuse HF du FT-707 en tenant compte de la
fréquence porteuse FI

   if (cw_blu)                          // si mode cw-am
   {
     frequaf = (freq - F_cw_am);        //freq = F(DDS)
     lcd.setCursor(0, 0);
     lcd.print("cw ");
     lcd.setCursor(0, 1);               // affichage du mode cw-am
     lcd.print("/am");
   }

  else if (LSB)                         // sinon mode LSB
   {
     frequaf = (freq - F_lsb);          // freq = F(DDS)
     lcd.setCursor(0, 0);
     lcd.print("lsb");                  // affichage du mode LSB
     lcd.setCursor(0, 1);
     lcd.print("    ");
   }

   else                                 // sinon mode USB

   {
     frequaf = (freq - F_usb);          // freq = F(DDS)
     lcd.setCursor(0, 0);
     lcd.print("usb");                  // affichage du mode USB
     lcd.setCursor(0, 1);
     lcd.print("    ");
   }

   frequaffich = frequaf/1000;          // valeur en kHz
   if (frequaffich > 0)                 // si 1kHz et plus
   {
    lcd.setCursor(4, 0);
    lcd.print (frequaffich);            // affichage des kHz
    lcd.print (".");                    // plus point décimal

    frequaffich = frequaf % 1000;                   // = reste des kHz
    if ((frequaffich < 10) && (frequaffich < -1))   // s'il n'y a que des unités
    {
     lcd.print("00");                                // alors, précédées de deux zéros
     lcd.print (frequaffich);
```

```
    }
    if ((frequaffich > 9)&& (frequaffich < 99))      // si unités et dizaines
    {
      lcd.print("0");                                // alors précédés d'un seul zéro
      lcd.print (frequaffich);
    }
    if (frequaffich > 99) lcd.print (frequaffich);  // sinon pas besoin de zéro(s)
  }
  else lcd.print (freq);
  if (freq > 999) lcd.print("KHz");                 // cas général en HF
  else lcd.print("Hz");                             // cas possible si F_min < 1 kHz
}

// FIN du programme
```