```
/**
 * Rotary encoder tuned for Raduino
 * Third build. Experimental BITX
 * V 1.0.6 ND6T  31 January 2019
 * Compiles under Etherkit Si5351 library v 2.0.6
 * This source file is under General Public License version 3.0
 * Corrected power formula.
 * Pin Connections:
 *                     A0 Plug 1 pin 8  black
 *      Forward RF     A1 Plug 1 pin 7  brown
 *      Reflected RF:  A2 Plug 1 pin 6  red
 *      S Meter:       A3 Plug 1 pin 5  orange
 *      Speed          A6 Plug 1 pin 2  blue
 *      Battery volts  A7 Plug 1 pin 1  violet
 *      T/R switching= D0
 *                  =  D1
 *      Encoder A =    D2 Plug 3 pin 6
 *      Encoder B =    D3 Plug 3 pin 5
 *      Encoder switch=D4 Plug 3 pin 4
 *      Key tip(dot)=  D5 Plug 3 pin 3
 *      Key ring(dash)=D6 Plug 3 pin 2
 *      Sidetone=      D7 Plug 3 pin 1
 */
#include <Rotary.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,10,11,12,13);
#include <si5351.h>
Si5351 si5351;

  Rotary r = Rotary(2,3); //Encoder to pins 2,3
  byte result;
  int ind = 4;            //Tuning position indicator
  int oldind;             //Indicator change reference
  int offset=700;         //CW offset, Sidetone frequency
  int wpm;                //Keyer speed
  int p ;                 //Timing period (milliseconds) for keyer function
  long count;             //Counter
  long lowerLimit = 7e6;  //Lowest frequency
  long upperLimit = 7.3e6;//Highest frequency
  long incr = 1000;       //Initial tuning increment
  long BFO= 11998488;     //Measured BFO frequency
  long LO = BFO -7.15e6;  //Initial frequency
  long oldLO;             //Old LO change reference
  float QSK=1.5;          //Delay (in seconds)for semi-QSK
  float sm=0;             //"S" meter value
  float FQ;               //Operating frequency
  float F=0;              //Forward RF output (PEP watts RMS)
  float FP;               //SSB ""
  float R=0;              //Reverse RF output (PEP watts RMS)
  float RP;               //SSB ""
  long long post;         //Time post

void setup(){
  PCICR |= (1 << PCIE2);                    //Interrupt setup
  PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);//Matrix "state machine" decode
  r.begin();   //Users that downloaded Rotary library before Dec.2018 should delete this line

  lcd.begin(16, 2);//Format and clear display
  lcd.clear();

  si5351.init(SI5351_CRYSTAL_LOAD_8PF,25004586L,0); //Ref osc freq.
  si5351.set_pll(SI5351_PLL_FIXED, SI5351_PLLA);
  si5351.set_freq(LO * 100, SI5351_CLK2);   //Program the synthesizer LO
```

```
  pinMode(0, OUTPUT);       //T/R keying for CW
  pinMode(4, INPUT_PULLUP); //Encoder switch
  pinMode(5, INPUT_PULLUP); //Dot Key
  pinMode(6, INPUT_PULLUP); //Dash Key
  pinMode(7, OUTPUT);       //Sidetone

  lcd.setCursor(0,0);////////////Splash//////////
  lcd.print("EZall");
  lcd.setCursor(0,1);
  lcd.print("version 1.0.6");
  delay(2000);
  post=millis();
  }

void loop(){
   FP=analogRead(A1)/(27e3/analogRead(A1)+1); //Read Forward RF power
   if(FP>F)F=FP;
   RP=analogRead(A2)/(27e3/analogRead(A2)+1); //Read Reverse RF power
   if(RP>R)R=RP;

   if(ind!=oldind){  //Display change of indicator position
    oldind=ind;
    show();
   }
   if(LO!=oldLO){   //If frequency changed then reprogram
   oldLO=LO;        //Reset reference
   program();       //Re-program the LO
   }

 if(((digitalRead(6)==LOW)&&(analogRead(A6)>=300)) || (digitalRead(5)==LOW)) CW();//Is the key
active?
 digitalWrite(0,LOW);  //Switch to Receive

 sm=(analogRead(A3));  //Read S meter.

 ind=(int(log10(FQ)))-(int(log10(incr)))+1;//Calculate indicator position
    if(incr>100)ind-=1;                    //Compensate for decimal place

 if ((millis() - post)> 2000) {  //If idle, display each 2 sec
    show();
    F=0;              //Reset forward power
    R=0;              //Reset reverse power
    post=millis();    //Reset time post
   }
}
//*****************FUNCTIONS (subroutines)***********************************

void show(){          //Display routine
  FQ=BFO-LO;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(FQ/1000,3);//Parse the display for easy reading
  lcd.print(" KHz");
  lcd.setCursor(14,0);
  if(analogRead(A6)<300){
      lcd.print("SK");}
      else {
        lcd.print((analogRead(A6)/30));//Display keying speed
        lcd.print(" WPM");
        }
  if(F>=1){            //If RF power present display measurements.
    lcd.setCursor(0,1);
    lcd.print(F,1);
    lcd.print("W  SWR=");
    lcd.print((1+sqrt(R/F))/(1-sqrt(R/F)),1);//Compute VSWR
    lcd.print(":1   ");
```

```
      }
  else{                //Otherwise display S meter
      lcd.setCursor(0,1);
      if (sm>=800)lcd.print("S9+50");
      if (sm>=700&&sm<800)lcd.print("S9+40");
      if (sm>=400&&sm<700)lcd.print("S9+30");
      if (sm>=300&&sm<400)lcd.print("S9+20");
      if (sm>=200&&sm<300)lcd.print("S9+10");
      if (sm>=170&&sm<200)lcd.print("S9");
      if (sm>=150&&sm<170)lcd.print("S8");
      if (sm>=100&&sm<150)lcd.print("S7");
      if (sm>=90&&sm<100)lcd.print("S6");
      if (sm>=70&&sm<90)lcd.print("S5");
      if (sm>=50&&sm<70)lcd.print("S4");
      if (sm>=40&&sm<50)lcd.print("S3");
      if (sm<40)lcd.print("S0");

  lcd.setCursor(10,1);
  lcd.print(analogRead(A7)/50.8);//Display supply voltage
  lcd.print("V");
  }
  lcd.setCursor(ind,0);//Indicator position
  lcd.cursor();        //Tuning increment indicator
  delay(50);           //Prevent flicker
}

void program(){
 si5351.set_freq(LO * 100, SI5351_CLK2);//Program the synthesizer
 show();                                //Update display
}

void CW() { //CW modes
  digitalWrite(0,HIGH);    // Key T/R relays and do the setup while they activate
  wpm = analogRead(A6)/30; //Read CW speed pot and set WPM rate
  p = 1200/wpm;            // convert speed to milliseconds

  if (wpm < 10)sk();   // Read speed control to switch to Straight Key mode
  if (wpm < 10)return; //To prevent race condition

  //Iambic keyer
  while (count < (QSK*5e4)) { // Delay time after last action to return to normal SSB
    if(digitalRead(6)==LOW)dah();
    if(digitalRead(5)==LOW)dit();
    count++;}            //Increment time-out for CW routine
    count=0;             //Reset the CW timeout
    digitalWrite(0, LOW);//Restore T/R relays from CW mode
    delay(100);          //Suppress relay click
    }

void dit() {   //Send a dot and an element space
   si5351.set_freq(((BFO-LO)-offset) * 100 , SI5351_CLK1); //Key on CW transmit frequency
    tone(7,offset); //Sidetone on D7
    delay(p);
    noTone(7);
   si5351.output_enable(SI5351_CLK1, 0);    // Unkey transmit
    delay(p);
   count=0;  //Reset counter
}

void dah() {  //Send a dash and an element space
   si5351.set_freq(((BFO-LO)-offset) * 100 , SI5351_CLK1); //Key on CW transmit frequency
    tone(7,offset); //Sidetone on D7
    delay(3*p);
    noTone(7);
    si5351.output_enable(SI5351_CLK1, 0); // Unkey transmit
    delay(p);
```

```
   count=0;  //Reset counter
}

void sk() {  //Straight Key mode
  while (count < (QSK*1000)) { // Delay time after last action to return to normal SSB
    if(digitalRead(5)==LOW)post=millis();     //Set post for display timing
    while(digitalRead(5)==LOW){
      si5351.set_freq(((BFO-LO)-offset) * 100 , SI5351_CLK1); //Key down
      tone(7,offset); //Sidetone
       if(millis()-post>1000){ //If keyed for more than a second, read power
         F=analogRead(A1)/(27e3/analogRead(A1)+1); //Read Forward RF power
         R=analogRead(A2)/(27e3/analogRead(A2)+1); //Read Reverse RF power
         show();
        }
      count=0; //Reset counter
    }

    si5351.output_enable(SI5351_CLK1, 0); // Unkey transmit
    noTone(7);
    count++;
    }
    count=0;             //Reset the CW timeout
    digitalWrite(0, LOW);//Restore T/R relays from CW mode
    }

ISR(PCINT2_vect) {               //Interrupt service routine
  result = r.process();
  if(digitalRead(4)==HIGH){      //If tuning knob is not pressed
   if(result == DIR_CW){
     LO-=incr;                   //Clockwise subtract the increment
     if(BFO-LO>upperLimit)LO=BFO-upperLimit; //Unless it exceeds upper limit
  }
  if(result == DIR_CCW){
    LO+=incr;                     //CounterClockwise add it.
    if(BFO-LO<lowerLimit)LO=BFO-lowerLimit; //Unless it is less than lower limit
    }
  }
   else{                 //If the tuning knob is pressed then move the cursor
   if(result == DIR_CW){ //Move cursor right
     incr=incr/10;
        if(incr<1)incr=1;//Lower limit
     }
   if(result == DIR_CCW){//Move cursor left
      incr=incr*10;
        if((log10(incr))>((log10(FQ))-1))incr=incr/10;//Upper limit
     }
  }
 post=millis();           //Display change
}
```