```
/**
 * Simple rotary encoder tuned, Arduino Nano, I2C SSD1306 OLED, Si5351
 * low level AD8307 measured variable level output
 * 4 KHz to 150 MHz
 * V 1.0.0 ND6T  8 January 2020
 * This source file is under General Public License version 3.0
 * Interrupt driven tuning. Potentiometer variable attenuator
 *
 * Pin Connections:
 *      Si5351 & Display: SDA = A4, SCL = A5
 *      Encoder: A = D2, B = D3, switch = D4
 */
#include <Rotary.h> //Available at: https://github.com/brianlow/Rotary
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
#include <si5351.h>  //Etherkit Si5351 library v 2.0.6
(https://github.com/etherkit/Si5351Arduino)
Si5351 si5351;

  Rotary r = Rotary(3,2);//Encoder to pins D2,D3
  unsigned char result;  //Encoder results
  int x=0;               //Utility variable
  int ind;               //Tuning position indicator
  int oldind;            //Previous tuning indicator
  int pos=2;             //Cursor position
  int cal=1140;          //Calibration value
  long incr = 1000;      //Initial tuning increment
  long upperLimit =15e7; //Upper frequency limit
  long oldFQ;            //Frequency change reference
  long long FQ = 10e6;   //Starting frequency
  float dbm=0;           //Result in dBm
  float uv=0;            //result in microvolts
  long long post;        //Time post for sensitive delays

#define sw 4    //Tuning increment switch

void setup(){
  PCICR |= (1 << PCIE2);//Interrupt setup
  PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);//Matrix "state machine" decode
  r.begin();

  display.begin(0x3C);  // initialize with the OLED I2C addr

  si5351.init(SI5351_CRYSTAL_LOAD_8PF,25000716L,0); //Ref osc freq.
  si5351.set_pll(SI5351_PLL_FIXED, SI5351_PLLA);
  si5351.drive_strength(SI5351_CLK2, SI5351_DRIVE_4MA);
  si5351.set_freq(FQ * 100, SI5351_CLK2);   //Program the synthesizer

  pinMode(sw,INPUT_PULLUP);//Tuning increment switch

  analogReference(INTERNAL);//Scale readings to 1.1 volts max.

 ////////////Splash//////////
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.print("EZGen3    V.1.0.0");
```

```
  display.display();
  delay(3000);
  post=millis();
  }

void loop(){
if(millis()-post<50) show(); //Display if changed less than 50 ms ago

// Read the input on analog pin 0:
for(int i = 0; i<1000; i++)dbm += analogRead(A0); //1000X oversample

  scale();               //and scale the results
  post=millis();         //Display results
  }

//*****************FUNCTIONS (subroutines)***********************************
void show(){    //Show 'em what you have!
  long fq=FQ;   //Re-cast to make it printable
  int mhz=(FQ/1e6);                //Truncate MHz
  long hz=FQ-(mhz*1000000);

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  if(mhz<10) display.print("0");

 if(FQ>=1e8)display.print(fq);          //If over 100MHz shorten display to fit
    else{
    display.print(mhz);                 //Parse at each 3 digits for easy reading
    if(FQ<1e8)display.print(" ");
     if((hz/1e3)<10) display.print("0");//Insert leading zeros if necessary
     if((hz/1e3)<100) display.print("0");
     display.print((hz)/1e3,3);         //Parse final 6 digits
    }
  display.setTextSize(2);
  display.setCursor((ind-1)*12,9);
  display.print("-");
  display.setCursor(0,17);
  display.print(uv);
//  display.print(analogRead (A0));
  display.setTextSize(1);
  display.print("microVolt");
  display.setCursor(70,25);
  display.print(dbm,0);
  display.print(" dBm");
  display.display();
  si5351.set_freq(FQ * 100, SI5351_CLK2);//Program the synthesizer
 }

void scale(){  //Scale it
  dbm=((dbm/1000)-cal)/10.5;        //Convert reading to dBm
  dbm=dbm-60;
  uv=(pow(10,(dbm/20)))*(100000*(sqrt(5)));    //Convert dBm to microvolts
}

ISR(PCINT2_vect){      //Tuning Interrupt service routine
 result = r.process();
  if(digitalRead(sw)==HIGH){        //If tuning knob is not pressed
```

```
  if(result == DIR_CW){
    FQ+=incr;                     //Clockwise. Add the increment
    if(FQ>upperLimit)FQ=upperLimit;//Unless it exceeds upper limit
 }
 if(result == DIR_CCW){           //CounterClockwise subtract
     if(((FQ)-incr)>=4000)FQ-=incr;//But keep it above 4 KHz
     }
 }
 else{     //If the tuning knob is pressed then move the cursor
     if(result == DIR_CW){ //Move cursor right
       if(pos<8)++pos;
       }
     if(result == DIR_CCW){ //Move cursor left
       if(pos>1)--pos;
       }
 }
 ind=pos;               //Correlate indicator to position

 if(FQ<1e8){            //If under 100 MHz
   if(ind>2)ind=pos+1;  //Compensate for decimal place
   if(ind>6)ind=pos+2;

 if(ind>9)incr=1;      //Set increment by cursor position.
 if(ind==9)incr=10;
 if(ind==8)incr=100;
 if(ind==6)incr=1e3;
 if(ind==5)incr=1e4;
 if(ind==4)incr=1e5;
 if(ind==2)incr=1e6;
 if(ind==1)incr=1e7;
 }
   else{                //If over 100 MHz
 if(ind==9)incr=1;      //Set increment by cursor position.
 if(ind==8)incr=10;
 if(ind==7)incr=100;
 if(ind==6)incr=1e3;
 if(ind==5)incr=1e4;
 if(ind==4)incr=1e5;
 if(ind==3)incr=1e6;
 if(ind==2)incr=1e7;
 if(ind==1)ind=2;
   }

if(oldind!=ind){
  oldind=ind;
  post=millis();
  }
}
```